IntechOpen

Tabu Search

Edited by Wassim Jaziri

Local Search Techniques: Focus on Tabu Search

Edited by

Wassim Jaziri

Tabu Search
http://dx.doi.org/10.5772/65
Edited by Wassim Jaziri

Notice

Statements and opinions expressed in the chapters are these of the individual contributors and not necessarily those of the editors or publisher. No responsibility is accepted for the accuracy of information contained in the published chapters. The publisher assumes no responsibility for any damage or injury to persons or property arising out of the use of any materials, instructions, methods or ideas contained in the book.

First published in Croatia, 2008 by INTECH d.o.o.
eBook (PDF) Published by IN TECH d.o.o.
Place and year of publication of eBook (PDF): Rijeka, 2019.
IntechOpen is the global imprint of IN TECH d.o.o.
Printed in Croatia

Legal deposit, Croatia: National and University Library in Zagreb

Additional hard and PDF copies can be obtained from orders@intechopen.com

Tabu Search
Edited by Wassim Jaziri

p. cm.

ISBN 978-3-902613-34-9

eBook (PDF) ISBN 978-953-51-5831-8

Preface

In real-world optimization applications, stakeholders require multiple and complex constraints, which are difficult to satisfy and make complicated to find satisfactory solutions. In the most cases, we face over-constrained optimization problems (no satisfactory solution can be found) because of the stakeholders' multiple requirements and the various and complex constraints to be satisfied. Solving over-constrained problems is based on the relaxation of some constraints according to values of preferences in order to favour the satisfaction of the most relevant. In addition, some methods have been developed, such as Branch and Bound method and Filtering techniques, which allow avoiding an enumeration of all potential solutions but are very expensive in computation times.

Two types of actors are involved in the optimization problems: the stakeholders who express their needs and the analyst who models and manages these needs. The choice of appropriate resolution methods depends on the stakeholders' needs and the number of criterion to take into account. The definition and the formulation of stakeholders needs are among the major preoccupations to resolve an optimization problem. The quality of a solution depends on the efficiency of this step, which must model all aspects of the problem, in particular those related to the objectives of the application. However, in majority of optimization works, the interest concerns the choice and the development of appropriate optimization approaches rather than the modelling of the stakeholders' needs.

An optimisation approach consists in exploring the search space to find the best solution according to an objective function and satisfying some constraints. It expresses a value of cost to reduce or of benefit to maximize. The optimization framework is adapted to the problems with multiple solutions. In traditional optimization problems, the objective function comprises a single criterion able to determine the optimum. Otherwise, the objective consists in finding a good compromise regarding multiple criteria. The problem is then a case of multicriteria optimization.

In practice, optimization problems can reach a high complexity and require considerable computation times because of the number of potential solutions. The approximate methods are generally used to resolve this class of problems. These methods are based on an iterative exploration of the search space to find a solution of good quality in reasonable computation times. Among the most known approximate methods, we mention the neighbourhood methods, such as Local Search, Simulated Annealing, Threshold Algorithms, Noising Method and Tabu Search as well as the algorithms based on the

evolution approach, such as Evolutionary Programming, Strategies of Evolution, Genetic Algorithms and Genetic Programming.

In the last decade, optimization problems have been among the most studied problems and they are still an active area of research. Algorithmic advances as well as the needs to solve complex real-world optimization problems have provided an excellent framework on which to develop and design new optimization techniques. The field of Neighborhood search methods has grown considerably. Researchers have demonstrated the ability of these methods to solve hard combinatorial problems within reasonable computation times.

Neighborhood search methods are iterative procedures, which consist in constructing from a current solution a next solution with a better quality regarding an objective function. Among the Neighborhood search methods, Tabu Search is one of the most prominent, widely used, and successful approaches for solving optimization problems from artificial intelligence and operations research. Tabu Search is a meta-heuristic algorithm, which can be used for solving combinatorial optimization problems. Tabu Search has found its usefulness in a vast area of applications such as scheduling, vehicle routing problems, resources planning, regional development, location problems, integer programming problems, traveling salesman, graph coloring, knapsack problems, network design etc. Tabu Search has now become a reputable optimization technique and has proved highly effective in solving a wide range of optimization problems. Several works presenting applications of Tabu Search to various combinatorial problems have proved that Tabu Search provides good solutions very close to optimality. These successes have made Tabu Search extremely popular among those interested in finding good solutions to the large combinatorial problems encountered in many practical settings.

Tabu Search uses a local search procedure to iteratively move from a current solution x to a neighbor solution x' in the neighborhood of x, until some stopping criterion has been satisfied. The search process starts with an initial solution and moves from neighbor to neighbor as long as possible while improving the objective function value. Tabu Search improves the performance of the search process by using memory structures: the last explored configurations are kept in a short-term memory (Tabu list) in order to prohibit moves that lead to one of them.

Overall objective of the book

The goal of this book is to report original researches on algorithms and applications of Tabu Search to real-world problems as well as recent improvements and extensions on its concepts and algorithms. The book' Chapters identify useful new implementations and ways to integrate and apply the principles of Tabu Search, to hybrid it with others optimization methods, to prove new theoretical results, and to describe the successful application of optimization methods to real world problems. Chapters were selected after a careful review process by reviewers, based on the originality, relevance and their contribution to local search techniques and more precisely to Tabu Search.

Audience

The book "Local Search Techniques: Focus on Tabu Search" provides a broad spectrum of advances in applied optimization with a focus on Tabu Search. It is designed to be useful and accessible to researchers, engineers, graduate students and all scientists and practitioners in computer science, operations research and artificial intelligence as well as

other applications specialists who need local search techniques to model and solve combinatorial optimization problems.

Preview

This book consists of two main parts: Tabu Search algorithms and applications. In the first part, the state of the art and novel advances of Tabu Search algorithms are described. The purpose of the applications section is to provide the practitioners with a description of the relevant optimization issues in a number of specific application areas. This book is organized as follows.

Chapter 1 "Tabu Search: a Comparative Study" provides an overall view of the "local search" area and describes Tabu Search and some other meta-heuristics to which Tabu Search will be compared: Simulated Annealing, Genetic Algorithms, Ant Colony Optimization, Greedy Randomized Adaptive Search Procedure and Particle Swarm Optimization. The authors identify the different problems for which Tabu Search was used to generate solutions, such as scheduling problems, routing problems, and assignment problems.

Chapter 2 "A Multiobjective Tabu Framework for the Optimization and Evaluation of Wireless Systems" provides an insight on the use of Tabu Search for the resolution of multiobjective optimization problems and its performance for real-world optimization problems. The main concepts of multiobjective optimization are presented as well as an overview of the main multiobjective strategies.

Chapter 3 "SOS-Heuristic for Intelligent Exploration of the Search Space in CSOP" proposes an intelligent search heuristic called *SOS-Heuristic* (Heuristic for Satisfactory and Optimized Solutions) to guide the resolution of constraints satisfaction and optimization problem. The objective of the SOS-Heuristic is to improve the value of the objective function without damaging the satisfaction of constraints. Experiments on a spatial optimization problem using a hybrid method provided interesting results and prove the efficiency of the proposed approach in comparison to another optimization approach based on Tabu Search.

Chapter 4 "Symbiotic Tabu Search" explains the use of the Tabu Search metaheuristic and proposes a hybrid Tabu Search approach with symbiotic evolution, called *Symbiotic Tabu Search*. Implementation and test results on three Benchmarks problems are presented.

In chapter 5 "Tabu Search and Hybrid Genetic Algorithms for Quadratic Assignment Problems", experience with solving quadratic assignment problems is reported. Six different Tabu Search methods are described.

Several improvement schemes for hybrid genetic algorithms are presented to help researchers who work on other problems as well to improve the performance of their genetic or hybrid genetic algorithms. Summary tables of computational experiments with various techniques are also presented.

Chapter 6 "A Hybrid GA-TS Technique with Dynamic Operators and its Application to Channel Equalization and Fiber Tracking" presents the author's research activities on the fields of Evolutionary Computation-based techniques applied to digital communications and medical image processing. The author presents a concise introduction to metaheuristic evolutionary computation-based strategies, mainly Genetic Algorithms and Tabu Search. A hybrid method based on Genetic Algorithm and Tabu Search is developed and applied on application examples.

Chapter 7 "Hybrid Tabu Algorithm for the Synthesis and Fabrication of Fiber Bragg Gratings" describes the use of a hybrid Tabu algorithm for the synthesis and fabrication of

fiber bragg gratings-based filters with specified frequency responses. The hybrid Tabu algorithm is a two-tier search that employs the global staged continuous Tabu Search algorithm and a local optimization algorithm.

Chapter 8 "Some New Results on Tabu Search Algorithm Applied to the Job-Shop Scheduling Problem" discusses the job shop scheduling problem and its representation with a disjunctive graph. The authors propose two innovative approaches for the job-shop scheduling problem. The first approach is based on a new neighborhood structure. The second approach combines Tabu Search and Simulated Annealing. Implementations and test results on benchmark problems are presented and compared with other approaches.

Chapter 9 "Tabu Search Experience in Forest Management and Planning" presents experiences with Tabu Search in forest planning efforts. The principles of Tabu Search are presented and a number of forest planning meta-models involving Tabu Search have been briefly described.

In chapter 10 "Feature Selection using Intensified Tabu Search for Supervised Classification", the authors discuss Tabu Search based algorithms for feature selection problems and compare them with other sequential feature selection algorithms. An overview about Tabu Search and Fuzzy objective function is proposed. To demonstrate the performance of the classification system using Tabu Search, a number of experiments have been made.

Chapter 11 "Hybrid Tabu Search for Re-Entrant Permutation Flow-Shop Scheduling Problem" applies the hybrid Tabu Search to solve re-entrant permutation flow-shop scheduling problems. The hybrid Tabu Search is compared to the optimal solutions generated using the integer programming technique and to the near optimal solutions generated by Tabu Search.

Chapter 12 "On the Design of Large-scale Cellular Mobile Networks Using Tabu Search" proposes a Tabu Search method to design large-scale mobile networks and to specifically solve the problem of assigning node base station to radio network controllers in cellular mobile networks. Experiments were conducted to measure the quality of solutions provided by the proposed algorithm and a comparison of the Tabu Search method to Genetic Algorithm and Simulated Annealing has been discussed.

Chapter 13 "Multiple Tabu Search Algorithm for Solving the Topology Network Design" presents the principle of Multiple Tabu Search algorithm and it application to solve the topology network design problem with considering both economics and reliability. To examine the performance of the proposed Multiple Tabu Search method, the authors compared the results with those of Genetic Algorithm, Tabu Search and Ant Colony Optimization.

Chapter 14 "Hybrid Approaches Tabu Learning Algorithm Based Neural Network" shows the efficiency of integrating Tabu Search into the neural network learning. It describes the typical artificial neural network including the architecture and the learning algorithm and proposes a Tabu-based neural network learning algorithm to solve the local convergence of the gradient method. The most basic neural network learning algorithm and the Tabu-based neural network learning algorithm are tested to approximate six different nonlinear functions.

Acknowledgements

Editing this book "Local Search Techniques: Focus on Tabu Search" would not have been possible without the most valuable input of a large number of people. First, I would

like to take the opportunity to thank I-TECH Education and Publishing for inviting me to edit this book. I also wish to thank all the authors for their contributions and the referees for the valuable help, comments and suggestions.

About the Editor

The Editor Dr. Wassim Jaziri born on 07.11.1975 received the B.S. and M.S. degrees in computer sciences from the University of Sfax, Tunisia, in 1998 and 1999 respectively, the Master Research (pre-doctorate diploma) in computer sciences from the University of Rouen, France, in 2000 and the Ph.D. degree in computer sciences from the National Institute of Applied Sciences (INSA) of Rouen, France, in 2004. In October 2004, he joined the French Agricultural Research Centre for International Development (CIRAD) of Montpellier (France), as a post-doctorate researcher. Actually, Dr. Jaziri is an assistant professor in the Higher Institute of Computer Science and Multimedia (ISIM) of Sfax-Tunisia and a member of the *Miracl* Laboratory (Multimedia, InfoRmation Systems and Advanced Computing Laboratory). His research interests include Optimization, Constraints satisfaction, Decision aid, Geomatics, Geographic Information Systems and Ontology.

September, 2008

Editor

Dr. Wassim Jaziri
Higher Institute of Computer Science and Multimedia,
University of Sfax,
Tunisia.

Contents

Part II: Tabu Search Applications

Part I:

Tabu Search Overview, Algorithms and Advances

1

Tabu Search: A Comparative Study

Harun Pirim, Engin Bayraktar and Burak Eksioglu
Mississippi State University, Industrial and System Engineering Department
USA

1. Introduction

Problems encountered in fields like scheduling, assignment, vehicle routing are mostly NP-hard. These problems need efficient solution procedures. If confronted with an NP-hard problem, one may have three ways to go: one chooses to apply an enumerative method that yields an optimum solution, or apply an approximation algorithm that runs in polynomial time, or one resorts to some type of heuristic technique without any a priori guarantee for quality of solution and time of computing (Aarts & Lenstra, 2003). Heuristics fall under the general heading of local search approaches. Hence, local search techniques are widely used to find "close-to-optimum" solutions to these problems in a "reasonable" amount of time. Tabu search (TS) is one of the most efficient heuristic techniques in the sense that it finds quality solutions in relatively short running time. This chapter will provide a basic description of TS giving insights for novice readers as well as introduce application areas and provide comparisons of TS to other meta-heuristic procedures for the readers with more experience on local search procedures.

The chapter will be organized as follows: The second section is going to introduce the basic terminology. For example, definitions for global optimization, local search, heuristics, and meta-heuristics will be provided. The section will also provide brief descriptions of TS as well as the following meta-heuristics to which TS will be compared: simulated annealing (SA), genetic algorithms (GA), ant colony optimization (ACO), greedy randomized adaptive search procedure (GRASP), and particle swarm optimization (PSO). Second section is intended to give the readers a good overall view of the "local search" area and let them know that TS will be compared to several other meta-heuristic procedures.

In the third section, basic steps of TS, SA, GA, ACO, GRASP and PSO will be described. As the mechanisms of these procedures are explained, differences and similarities between TS and each of the other procedures will be pointed out. Section three will familiarize the readers with the various meta-heuristic procedures that will be discussed throughout the chapter.

The fourth section will be dedicated to identifying the different problems for which TS was used to generate solutions. For example; TS has been used to solve scheduling problems, routing problems, and assignment problems. We will try to generate a comprehensive list of the problems to which TS has been applied. This section will provide the reader with an understanding of how TS has been used.

In the fifth section, efficiency and effectiveness of TS will be compared to other meta-heuristic procedures. Reasons why TS is more efficient and/or effective than some of the

other local search techniques will be discussed. Section five will explain why TS is the choice of solution method for some problems and not for others. Section six will conclude the chapter.

2. Definitions and terminologies

Tabu Search (TS) was developed by Fred Glover in 1988. It was initiated as an alternative local search algorithm addressing combinatorial optimization problems in many fields like scheduling, computer channel balancing, cluster analysis, space planning etc. (Glover, 1989). However, popularization and dissemination of TS goes back to the works of Hertz and de Werra (1987, 1989, 1991). This section consists of three parts: general definitions, TS related definitions, and definitions related to other meta-heuristics.

2.1. General definitions

The term "combinatorial" refers to the constraint that the solution set has to be finite or countably infinite (Michiels et al., 2007). Many combinatorial optimization problems can be expressed as a search for a specific permutation (Dréo et al. 2006). Solution space of combinatorial optimization problems can typically be represented by sequences, permutations, graphs and partitions (Michiels et al., 2007).

Combinatorial optimization problem: Optimizing a linear function subject to other linear functions over a finite (or countably infinite) set of possible solutions is called a combinatorial problem. Combinatorial optimization is the discipline of decision making in case of discrete alternatives (Aarts & Lenstra, 2003). In other words, in combinatorial optimization, one looks for an object from a finite, or countably infinite set, permutation, or graph (Papadimitriou & Steiglitz, 1998).

Global and local optimum: An optimization problem with a feasible solution set S and a neighborhood function N has a local optimal solution that is also globally optimum if N is exact. A solution is locally optimum if and only if its out degree is zero in the transition graph which is a directed, acyclic sub-graph of a neighborhood graph. A globally optimum solution can be found within a small number of steps if the neighborhood graph is strongly connected, which means for each pair of solutions (a, b), b is reachable from a, and its diameter (maximum distance between any pair of solutions) is not too large. If a graph is not strongly connected then its diameter is infinitely large. A local optimal solution to a problem may be poor (i.e. far from the global optimum). Hence, a better solution can be generated by applying a more powerful neighborhood function which obviously is a trade-off between quality of a solution and computation time to yield that solution.

Complexity: "A measure of computer time or space to solve a problem by an algorithm as a function of the problem's dimensions. Suppose $T(n)$ is the time it takes to solve an instance of a problem with dimension n. Then, the algorithm has (worst-case) time complexity K(n), if the greatest time it could take to solve an instance of the problem is O(K(n)). When K(n) is a polynomial, we say the algorithm has polynomial time complexity"(Holder, 2006). If the running time of an algorithm is not polynomial then it is typically exponential. For example, if we try to find the best tour for a Travelling Salesman Problem (TSP) with one hundred cities, the number of solutions exceeds 10^{50} which is larger than the estimated number of particles in the universe (Michiels et al., 2007). If a problem is polynomially reducible to

another problem then the new problem is at least as hard as the old one and a polynomial-time algorithm exists for the new problem if and only if it exists for the old problem.

Heuristics, meta-heuristics, hyperheuristics: Heuristic usually refers to a procedure that seeks an optimum solution but does not guarantee it will find one, even if one exists. Meta-heuristics are general frameworks for heuristics in solving hard problems. The idea of "meta" is that of level (Holder, 2006). Meta-heuristics do not stop in the first local optimum as a simple heuristic does. They can be classified into two: those that perform a single walk in the neighborhood graph using special procedures trying not to be trapped in a local optimum and those that perform multiple walks (Michiels et al., 2007). TS and SA are examples for the first class. Hyperheuristics choose between given heuristics at various decision points in an optimization problem.

Constructive algorithm: An algorithm that generates a solution through a number of steps where in each step a partial solution is obtained and a complete solution is obtained in the final step.

Plateaus: A part of a solution space that contains solutions with the same objective function value.

Local search algorithm: An algorithm that searches through the solution space and tries to find good quality solutions in each step by means of a neighborhood.

Graph representation of solutions may be inspiring for the designer to be able to direct the search more intelligently (Dréo et al. 2006). For a local search algorithm to be effective, solution space of the problem should not comprise large plateaus. Plateaus may cause cycling. There are ways of avoiding cycling such as remembering recently visited solutions as in TS short term memory.

Local search is what we always do when we are supposed to find a solution in practical life as well. Local search associates by local optimum and local optimum may be a step/stop for global optimum. One may try to modify a local optimal solution in order to get a better solution. However, it is necessary to prevent cycling among solutions visited. This probability of revisiting a previously visited solution is inevitable unless necessary cautions are taken. In that sense, TS uses memory property to prevent cyclic motions in the solution space. TS uses short-term and/or long-term memory while making moves between neighboring solutions. It is essential for a local search to be balanced in terms of quality of solutions and computing time of these solutions. In that sense, a local search does not necessarily evaluate all neighborhood solutions. Generally, a subset of solutions is evaluated.

We can give a maze analogy to explain how local search works: a man needs to find the door to get out of the maze. All paths he travels look similar. He goes back and forth to find the exit door. He makes moves to be able to search through the maze. He has some signs-such as colorful marbles- which he can put on the floor of areas he walked through in order to understand that he visited these areas previously so that he can narrow his search space and easily find the door which will lead him outside. In this analogy, the maze represents the search space of neighborhood solutions of a problem instance. Moves made by the man are the moves (iterations) of an algorithm. His back and forth moves may represent cycles. Marbles are rules (e.g. tabu list) that prevent an algorithm from being trapped in a cycle or

local optimum and narrow the search space. The door opening outside represents a local or global optimal solution of a problem instance.

A local search algorithm begins with an initial solution. This initial solution can be generated by any heuristic algorithm. The algorithm then searches through the solution space with guidance of a neighborhood function. In other words, the algorithm makes a walk through the neighborhood graph. There are different strategies for walking through a neighborhood graph. The most obvious strategy is used by the iterative improvement algorithm also known as the hill climbing algorithm (Michiels et al., 2007). This basic algorithm searches for a better solution in the neighborhood. If it finds a better solution, it changes the current solution with this new one. Otherwise, the algorithm stops and keeps the current local optimum solution (Figure 2.1). The simplex method of linear programming is also a hill climbing procedure that moves from one extreme point solution to another, using an exact neighborhood.

Algorithm Hill Climbing (Iterative improvement)
begin
i:=initial solution
repeat
generate an *s* ϵ *N(i);*
if f(*s*) > f(*i*) **then** *i*:=*s*;
until f(*s*) ≤ f(*i*) for all s ϵ *N(i);*
end;

Figure 2.1 Hill climbing algorithm

In iterative improvement, selecting a better solution within a neighborhood is done using what's called a pivoting rule. A pivoting rule generally selects the first better solution or the best solution within a neighborhood. We had mentioned that we could represent the solution space as a transition graph. The potential of such a graph gives a lower bound on the number of iterations that are maximally required by iterative improvement. One disadvantage of applying iterative improvement is the possibility of being trapped at a local optimum. In order to escape from such a possibility, the neighborhood function can be defined accordingly. Another alternative is allowing non-improving moves as well as performing multiple runs of iterative improvement (Michiels et al., 2007). For example, TS and SA allow non-improving moves as it will be discussed in more detail in the following sections.

Neighborhood function: Given a feasible solution *s*, we can define a set of solutions *N(s)* elements of which are close to *s*. Neighborhood functions can be generated based on the structure of the problem at hand. Some basic neighborhoods are inversions of two elements placed successively in the permutation, transposition of two distinct elements, or displacement of an element (Dréo et al. 2006). If a solution *s* is neighbor of solution *s′* and *s′* is also a neighbor solution of *s* then *N i*s called a symmetric neighborhood function. Neighborhood functions can be defined as swapping, moving, replacing operations. A neighborhood is said to have a performance bound *C* if all local optimum costs have at most a cost of *C* times optimum cost. If the neighborhood function is exact then iterative improvements end up with an optimal solution.

2.2 Tabu search definitions

Short-term memory: A kind of memory that is limited in terms of time and storage capacity. In TS, the tabu list can be regarded as a short-term memory. Recency memory which will be defined later is also a short-term memory. With a short term memory, a previsited solution may be revisited with a different neighborhood.

Long-term memory: A kind of memory that differs from short-term memory in terms of time and storage capacity. The probability of visiting a previously visited solution using long-term memory is very small. Intensification and/or diversification are/is achieved through long-term memory.

Move: A modification made to a solution. Local search will tend to visit previously visited solutions more frequently if the number of tabu moves are relatively small. On the other hand, if the number of tabu moves is large then the search is less probable to find good local optima due to lack of available moves.

Tabu list: In order to prevent visiting a previously visited solution, TS uses a tabu list in which tabu moves or attributes of moves are listed. Also, tabu name is originated by these prohibited move states. Short tabu lists may not prevent cycling resulting in information loss while long tabu lists may excessively prevent neighborhood so that moves are limited to some extent.

Aspiration: Some tabu moves need to be disregarded in order to obtain better local solutions. These moves are called aspired moves. "Improved-best" and "aspiration by default criteria" are examples. Moving to a solution better than the last solution found so far is a commonly used aspiration criterion as well.

Probabilistic tabu search: Unlike many TS applications which have deterministic moves, probabilistic TS assigns a probality for each move (Glover & Laguna, 1997). Convergence theorems for SA can be adapted to probabilistic tabu conditions.

Quality dimension of memory: Ability to differentiate the merit of solutions to problem instance visited during the search (Glover & Laguna, 1997). Using a quality dimension, memory can be used to describe the elements of a good solution so that a bad move can be penalized.

Recency-based memory: A kind of short-term memory that keeps tracks of solution attributes that have changed during the recent past. This is the most described TS feature in the literature (Glover & Laguna, 1997).

Frequency-based memory: To make sure that a certain level of diversity is maintained throughout the search without prohibiting many moves, frequently used moves can be penalized (Dréo et al. 2006). Variety of penalizing methods can be derived. Frequency of certain moves not exceeding a predefined threshold is an example. It is obvious that a penalty mechanism must be used along with an aspiration criterion not to miss good solutions. Penalty based on frequencies may be regarded as a long-term memory. In general, it is better to use both long-term and short-term memory together. Such collaboration can be either "constant" or "varied". The term "constant" indicates that the number of tabu moves and penalty coefficients are predetermined. On the other hand, "varied" refers to alternating search phases. The purpose of these phases will be to intensify the search or to diversify the search. Intensification is related to reducing the number of prohibited moves and/or evoking the long-term memory to choose solutions with characteristics close to the best solutions enumerated by the search. Diversification is achieved to some extent by means of a short term memory.

Intensification aims to guide the search through a part of the solution space that is likely to have promising solutions while diversification aims to force the search through unvisited areas in the solution space.

Critical event memory: A kind of memory that monitors the occurrence of critical events and constitutes an aggregate summary of these events (Glover & Laguna, 1997).

Explicit memory: Records complete solutions generally consisting of elite solutions (Glover & Laguna, 1997). This kind of memory needs excessive space, and it expands neighborhood during the search. Explicit memory is related to intensification.

Attributive memory: Records information about solution attributes rather than about solution. In the TSP, index of tours may be used as attributes. Attributive and explicit memories complement each other. Attributive memory reduces the neighborhood size by forbidding certain moves.

Tabu-active: Attributes that change as a result of a move. These attributes may be used in a recency-based memory called "tabu-active" for a specified number of iterations.

Tabu tenure: Size of a tabu list or previous solutions to be stored. Effective tabu tenures depend on the size of a problem instance.

Strategic Oscillation: A means for interplay between intensification and diversification over a long term (Figure 2.2)

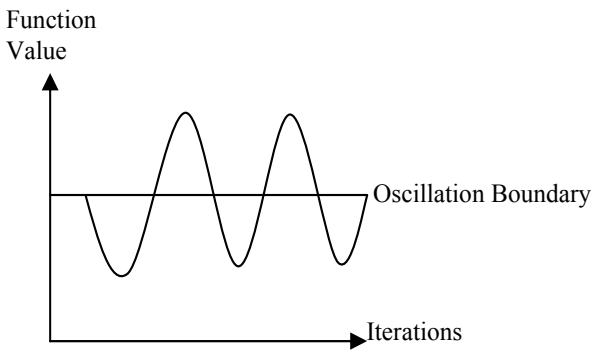


Figure 2.2 Graph of Oscillation Boundary

Tabu Thresholding: A method that joins prescriptions of strategic oscillation with candidate list strategies. Thresholding methods guide greedily exploring search space in a nonmonotonic way.

Hash functions: These are used in TS to provide a mechanism to avoid cycling in such a way that it won't result in expensive computation. Solution vectors can be mapped to integers that can be stored for a number of recent iterations.

Path relinking: A kind of integration of intensification and diversification strategies. Based on this approach, new solutions are generated exploring trajectories that connect elite solutions by generating a path in the neighborhood space (Glover & Laguna, 1997).

Vocabulary building: It is about generating new attributes out of other attributes like generating new populations or offsprings in GA. The basic idea is identifying meaningful parts of solutions as a basis to generate new combination of solutions. It can be viewed as an instance of path relinking.

2.3 Other meta-heuristic definitions

All meta-heuristics, including TS, have terminology and stories associated with them. For example, some meta-heuristics are inspired from animal behaviors, others from biology, and some from manufacturing processes. In this part of the chapter, we will concentrate on the definitions related to meta-heuristic procedures other than TS.

2.3.1 Simulated annealing (SA)

The SA procedure is inspired from the annealing process of solids. SA is based on a physical process in metallurgy discipline or solid matter physics. Annealing is the process of obtaining low energy states of a solid in heat treatment. Annealing process starts with melting the solid by heat treatment. Particles constituting the solid are arranged according to that heat treatment. Then, temperature is decreased which results in minimum energy state.

Threshold Value: Value from which difference of costs associated with two solutions should be strictly less.

Temperature (control parameter): It is the expected value of the thereshold.

2.3.2 Genetic algorithms

Chromosomes: These are strings of parameters which construct proposed solutions to the problem.

Crossover: In order to find better solutions and maintain diversity, crossover is used in genetic algorithms. For instance, a new solution (a child) can be obtained by combining two separate solutions (parents). This combination is defined as crossover. Figure 2.3 provides an example. There are two chess boards and pawns on these boards. We divide the chess boards from the middle into two parts. On the first chess board, we have 3 pawns on the left and 3 pawns on the right. On the second chess board, we have 6 pawns on the left and 4 pawns on the right. An example of crossover would be combining the left half of the first board with the right half of the second board. It is possible to obtain diverse solutions in genetic algorithms by utilizing the crossover operation. However, crossover may lead to infeasible solutions, and one needs to be aware of such situations.

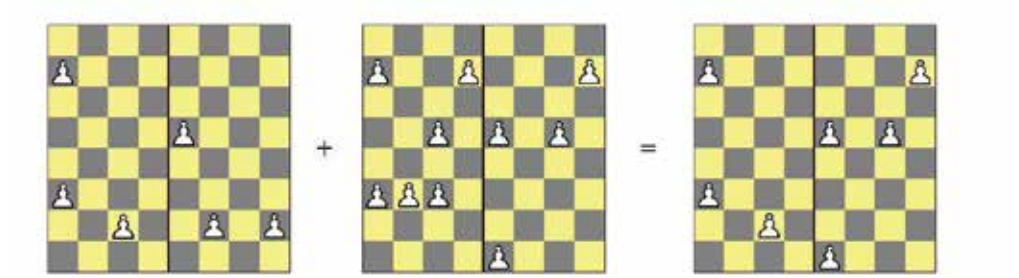


Figure 2.3 A crossover example considering different pawn orders on the chess board

Evolutionary Algorithms: Algorithms which are very similar to genetic algorithms. They are population based algorithms and they rely on artificial intelligence.

Fitness Function: A tool used in genetic algorithms that serves as a neighborhood function so that different solutions can be compared based on the values obtained from the fitness function.

Initialization: A step in which initial solutions are obtained by using initial populations. These initial populations are usually generated randomly. Using these populations, possible solutions are determined some of which are selected to construct the search space.
Mutation: An operator which is used to provide diversity in between populations.
Selection: This is the process in which the solutions are selected in accordance with the results from the fitness function.
Reproduction: After obtaining the results through crossover and/or mutation, another population is generated. From this newly created population, again new children are determined by the help of crossover and/or mutation and the reproduction process goes on.
Termination criterion: A condition that indicates when the algorithm will stop.

2.3.3 Ant colony optimization

Ants: Small animals (insects) that live in colonies in/on the ground. With this real life definition, ant colony optimization is an optimization method in which imaginery agents are used.
Daemon Actions: These are the actions that can be taken to centralize the solution. The aim of Daemon Actions is to prevent quick convergence of the algorithm.
Decentralized Control: A term which is related to robustness and flexibility. Robust systems are desired because of their ability to continue to function in the event of breakdown of one of their components (Dréo et al., 2006).
Dense Heterarchy: A term which is taken from biology and represents the organization of ant colonies. It is different from the managerial term hierarchy. In dense heterarchy, the structure is horizontal, contrary to hierarchy (see Figure 2.4).
Pheromone: In real life, pheromone refers to the chemical material that an ant spreads over the path it goes and the level of it changes over time by evaporating. On the other hand, in ant colony optimization, pheromone is a parameter. The amount of this parameter determines the intensity of the trail. The intensity of the trail can be viewed as a global memory of the system (Dréo et al., 2006).

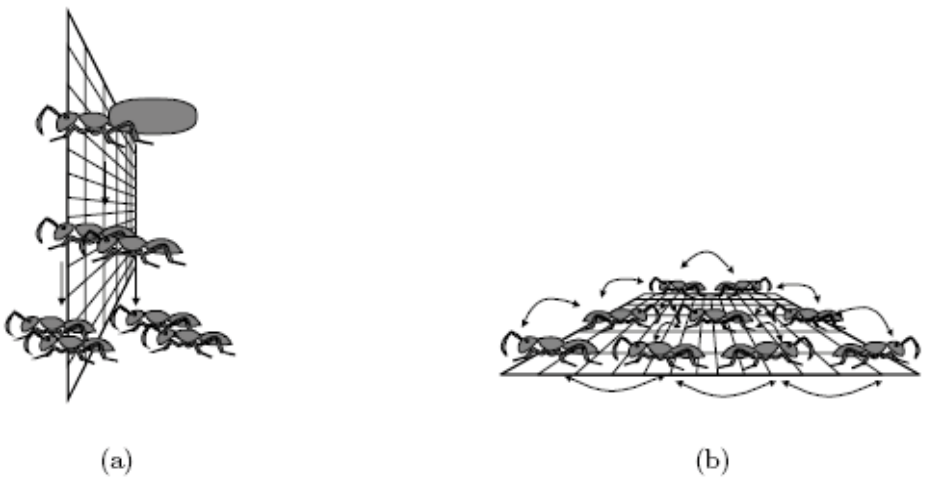


Figure 2.4 - Hierarchy (a) and dense heterarchy (b): two opposite concepts (Dréo et al., 2006)

MAX-MIN Ant System: An improved version of the "Ant System" in which only the best ant updates a trail of pheromone and values of the trails are limited, the maximum value is given to the trail initially (Dréo et al., 2006).
Positive Feedback: Feedback that instructs all ants to follow the same single path to reach the solution.
Stigmergy: The indirect communication among ants when finding a path to reach the food.

2.3.4 GRASP

Greedy Function: The function is used to rank the solution elements.
Iterated Local Search: It is the search in which a locally optimal solution is derived by using iterative improvement of GRASP.
Restricted Candidate List (RCL): A list in which well ranked elements of partial solutions are placed.

2.3.5 Particle swarm optimization

Cognitive Consistency: A meaningful pattern among the particles in a given society. Particles can be anything from animals to cities depending on the system studied. Based on cognitive consistency, when one group of particles thinks in a certain way, the other groups also think in the same or a similar way.
Global Best: The best position found after an update made by any particle in the swarm.
Local Best: The best solution that a particle has seen.
Neighborhood Best: It is the best solution that the particle finds after examining the neighboring solutions.
School: The set of elements (e.g. animals) in a society.
Position: The location of the particle in a specific iteration.
Social Influence: A term used in PSO that describes the logic of particle swarm. In real life, people have thoughts and these thoughts can change after social interactions like conversations. The same logic is used for PSO so that the solutions are changed in the best possible way.
Swarm Intelligence: The artificial intelligence that is made up of simple agents that interact with one another and with the environment.
Velocity: The direction of movement of the particles of a particular society.

3. Meta-heuristic procedures

It is possible to classify meta-heuristics in many ways. Different view points differentiate the classifications. Blum and Roli (2003) classified meta-heuristics based on their diverse aspects: nature-inspired (e.g. GA, ACO) vs. non-nature inspired (e.g. TS); population-based (e.g. GA) vs. single point search (also called trajectory methods, e.g. TS); dynamic (i.e. guided local search) vs. static objective function; one vs. various neighborhood functions (i.e. variable neighborhood search); memory usage vs. memory-less methods. A classification of meta-heuristics is given in the Table 3.1 in which "A" represents the adaptive memory property, "M" represents the memory-less property, "N" represents employing a special neighborhood, "S" represents random sampling, "1" represents

iterating-based approach, and "P" represents a population-based approach. Population-based approaches, also referred to as evolutionary methods, manipulate a set of solutions rather than one solution at a stage.

Meta-heuristic	Classification
Tabu-search	A/N/1-P
Simulated annealing	M/S-N/1
GA	M/S-N/P
ACO	M/S-N/P
GRASP	M/S-N/1
PSO	M/S-N/P

Table 3.1 – Classification of Meta-heuristics (modified from Glover, 1997)

Almost all meta-heuristic procedures require a representation of solutions, a cost function, a neighborhood function, an efficient method of exploring a neighborhood, all of which can be obtained easily for most problems (Aarts & Lenstra, 2003). It is important to mention that a successful implementation of a meta-heuristic procedure depends on how well it is modified for the problem instance at hand.

3.1 Tabu Search (TS)

TS can be considered as a generalization of iterative improvements like SA. It is regarded as an adaptive procedure having the ability to use many methods, such as linear programming algorithms and specialized heuristics, which it guides to overcome the limitations of local optimality (Glover, 1989).

TS is based on concepts that can be used in both artificial intelligence and optimization fields. Over the years TS was improved by many researchers to become one of the preferred solution approaches. Surrogate constraints, cutting plane approaches, and steepest ascent are big milestones in the improvement of TS. TS applies restrictions to guide the search to diverse regions. These restrictions are in relation to memory structures that can be thought of as intelligent qualifications. Intelligence needs adaptive memory and responsive exploration (Glover & Laguna, 1997). For example, while climbing a mountain one remembers (adaptive memory) attributes of paths s/he has traveled and makes strategic choices (responsive exploration) on the way to peak or descent. TS also uses responsive exploration because a bad strategic decision may give more information than a good random one to come up with quality solutions. TS has memory property that distinguishes it from other search designs. It has adaptive memory that is also different from rigid memory used by branch and bound strategies. Memory in TS has four dimensions: quality, recency, frequency, and influence.

TS forces a move to a neighbor with least cost deterioration. TS uses memory to keep track of solutions previously visited so that it can prevent revisiting that solution. Memory-based strategies are hallmark of TS approaches. Many applications don't include advanced features of TS since good solutions are typically achieved by simple designs. A basic tabu-search algorithm for a maximization problem is illustrated in Figure 3.1.

```
algorithm Tabu search
begin
    T:= [ ];
    s:=initial solution;
    s*:=s
    repeat
        find the best admissible s′ ϵ N(s);
        if f(s′) > f(s*) then s*:=s′
        s:=s′;
        update tabu list T;
    until stopping criterion:
end;
```

Figure 3.1 – A basic tabu search algorithm

where T is a tabu list and *N(s)* is the set of neighborhood solutions. A generic flowchart of TS algorithm can be given as follows in Figure 3.2:

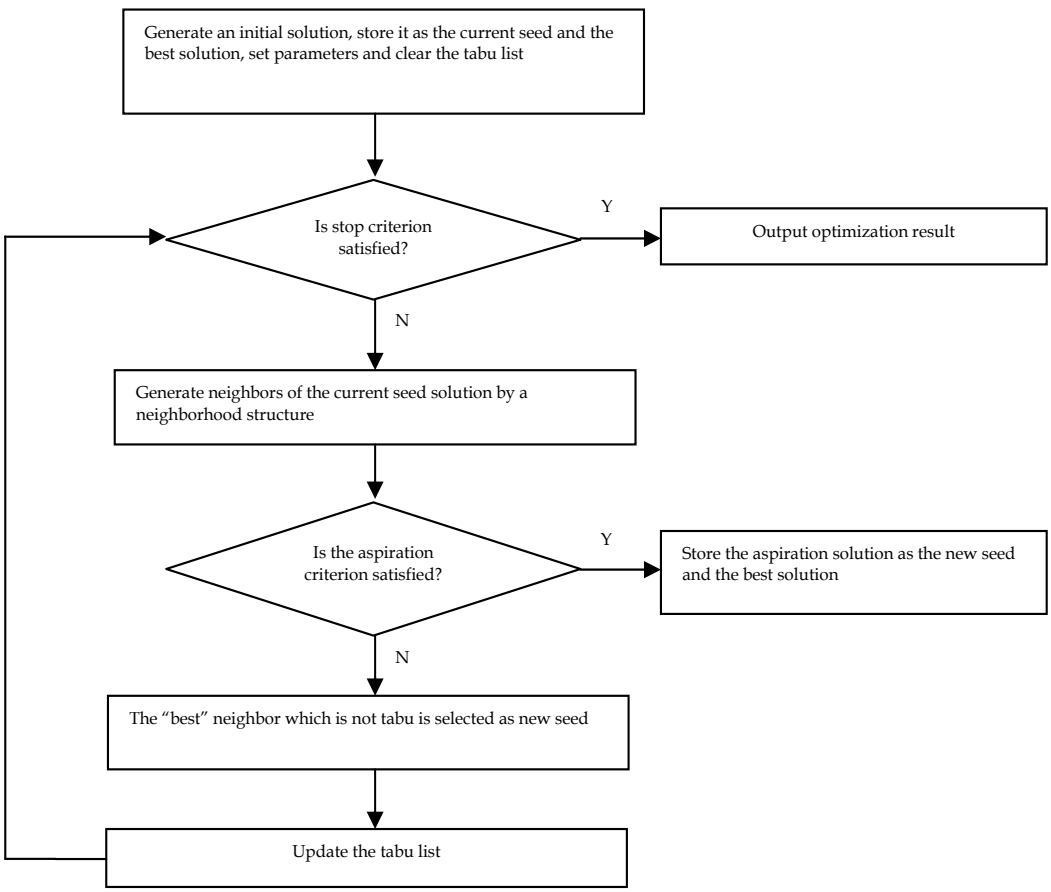


Figure 3.2 - Generic flowchart of TS algorithm (Zhang et al. 2007)

TS memory can be implemented by means of matrices as shown in the practical example below provided in Figure 3.3 (Hindsberger & Vidal, 2000) for the TSP:

Iteration 10

Recency M.

	1	2	3	4	5
1		3			
2	1				2
3				1	
4	2	1	3		
5		2			

Frequency M.

Current solution:

City	1	2	3	4	5
Seq.	1	3	2	4	5

Objective Value: 16

Best Objective Value: 15

Neighborhood:

	Swap	Obj. Value
T	2 - 5	12
	2 - 4	17
	3 - 5	18

Iteration 11

Recency M.

	1	2	3	4	5
1		2			
2	1				3
3					
4	2	1	3		
5		3			

Frequency M.

Current solution:

City	1	2	3	4	5
Seq.	1	5	2	4	3

Objective Value: 12

Best Objective Value: 12

Neighborhood:

Swap	Obj. Value
1 - 3	15
2 - 4	15
3 - 4	16

Figure 3.3 – Matrix implementation of recency memory (Hindsberger & Vidal, 2000)

Here, upper triangular matrix represents recency memory which stores tabu moves. For example, in iteration 10 exchanging cities 1 and 2 is tabu for 3 iterations, exchanging cities 2 and 5 is tabu for 2 iterations etc. Lower triangular matrix represents frequency memory which stores frequency of exchanging cities. For example, in iteration 10 cities 1 and 2 were changed once etc. Total number of exchanges is 9 since it is the 10^{th} iteration we are in. T represents the tabu move.

3.2 Simulated Annealing (SA)

SA is a randomized algorithm that tries to avoid being trapped in local optimum solution by assigning probabilities to deteriorating moves. In SA a threshold value is chosen. The increase in cost of two moves is compared with that threshold value. If the difference is less than the threshold value, then the new solution is chosen. A high threshold value may be chosen to explore various parts of solution space while a low threshold value may be chosen to guide the search towards good solution values. The threshold value is redefined in each iteration to enable both diversification and intensification. Starting with high threshold

values and then decreasing the value may result in finding good solutions. SA uses threshold as a random variable. In other words SA uses expected value of threshold. In a maximization problem acceptance probability of a solution is defined as follows:

$$IP\{s'\} = \begin{cases} 1 & f(s') \geq f(s) \\ \exp\left(\dfrac{f(s') - f(s)}{c_k}\right) & f(s') < f(s) \end{cases}$$

where c_k is the temperature that gives the expected value of the threshold. A generic SA algorithm for a maximization problem is given in Figure 3.4 below:

```
algorithm Simulated annealing
begin
    s:= initial solution
    k:=1;
    repeat
        generate an s′ ϵ N(s);
        if f(s′) ≥ f(s) then s:=s′
        else
            if exp( (f(s') − f(s)) / c_k ) > random[0,1) then s:=s′;
        k:=k+1;
    until stopcriterion:
end;
```

Figure 3.4 – A simulated annealing algorithm

The cooling schedule is important in SA. Temperature values *(c_k)* are specified according to the cooling schedule. In general, the cooling schedule's temperature is kept constant for a number of iterations before it is decreased.

3.3 Genetic Algorithms (GAs)

GAs are used to create new generation of solutions among trial solutions in a population. GA is a population-based heuristic that imitates a biological system to find a reasonable solution to a difficult problem. In this section, we will observe how one can obtain efficient solutions with respect to computation time and solution quality using GAs.

In a GA, a "fitness function" is utilized and hence a quantitative study is performed. The fitness function evaluates candidate solutions, determines their weaknesses and deletes them if they are not expected ones. After this step, the reproduction among the candidates occurs and new solutions are obtained and compared using the fitness function again. The same process keeps repeating for number of generations.

With the above description in mind, Figure 3.5 shows a general schema of using GA for minimization problems. The initial step is to determine P_0, the first population of solutions. Using the fitness function, improvements are made to the initial population of solutions. Afterwards, the algorithm enters into a loop in which crossover and mutation operations are performed until a stopping criterion is met. A typical stopping criterion is to perform all the steps for a fixed number of generations.

```
Begin
P0 := set of N solutions;
/*Mutation*/
replace each s ∈ P0 by Iterative_Improvement(s);
t :=1;
repeat
Select Pt ⊆ Pt-1;
/* Recombination */
extend Pt by adding offspring;
/* Mutation */
replace each s ∈ Pt by Iterative_Improvement(s) ;
t :=t+1;
until stopcriterion;
end;
```

Figure 3.5 - A genetic local search algorithm for a minimization problem (Michiels et.al., 2003)

GAs have many application areas in Aerospace Engineering, Systems Engineering, Materials Engineering, Routing, Scheduling, Robotics, Biology, Chemistry, etc.

3.4 Ant Colony Optimization (ACO)

ACO is another branch of meta-heuristics that is used to solve complex problems in a reasonable amount of time.
In Figure 3.6, a general type of ant colony optimization is given.

```
procedure ACO_Meta-heuristic
        while (not_termination)
        generateSolutions()
        pheromoneUpdate()
        daemonActions()
        end while
end procedure
```

Figure 3.6 - A general ant colony optimization procedure

As seen from the general algorithm, a set of initial solutions should be generated in each turn of the while loop, then the pheromone levels should be updated and actions should be taken. When the termination criterion is reached, the procedure ends. This algorithm can be modified to fit the needs of the specific problem.
In Figure 3.7, a generic ACO procedure is provided to solve the Traveling Salesman Problem (TSP). J_i^k is the list of already visited cities.

As shown in Figure 3.7, the algorithm begins with generating an initial solution. The variable m represents the number of ants. Each city is selected based on a probability function which is given in *(1)* and after the selection; the evaporation is performed by utilizing equation *(2)*. However, in order to calculate equation (2), equation (3) is utilized.

For Iteration t = 1, . . . , t_{max}
  For each ant k = 1, . . . , m
    Choose a city randomly
    For each non visited city i
      Choose a city j, from the list of J_i^k remaining cities using *(1)* given below.
    End For
    Deposit a trail $\Delta\tau_{ij}^k$ on the path $T^k(t)$ in accordance with *(3)* given below.
  End For
  Evaporate trails based on *(2)* given below.
End For

Figure 3.7 - An ACO algorithm used in solving the TSP

$$p_{ij}^k(t) = \begin{cases} \dfrac{(\tau_{ij}(t))^{\alpha}.(n_{ij})^{\beta}}{\sum_{t\in J_i^k}(\tau_{il}(t))^{\varepsilon}.(n_{ij})^{\beta}} & if \quad j \in J_i^k \\ 0 & if \quad j \notin J_i^k \end{cases} \tag{1}$$

where;

α and β are parameters that controls τ_{ij} and n_{ij}.

n_{ij} is the visibility from city i to city j.

τ_{ij} is the intensity from city i to city j.

ρ is evaporation coefficient.

where;

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{2}$$

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{Q}{L^k(t)} & if \quad (i,j) \in T^k(t) \\ 0 & if \quad (i,j) \notin T^k(t) \end{cases} \tag{3}$$

As the best solution is obtained, the algorithm stops. All the equations and algorithms are borrowed from Dréo et.al. (2006).

3.5 Greedy Randomized Adaptive Search Procedure (GRASP)

GRASP is another meta-heuristic method used for solving combinatorial optimization problems. Figure 3.8 demonstrates how GRASP works for a minimization problem.

This algorithm is composed of two main phases: a construction phase and a local search phase. In the construction phase, there is a greedy function which maintains the rankings of partial solutions. This step is very important because it affects the time efficiency of the algorithm. After ranking the partial solutions, some of the best ones are stored in a restricted candidate list (RCL). In the local search phase, as shown in Figure 3.8, a comparison is done

to differentiate the quality of solutions. The algorithm terminates after a fixed number of iterations.

```
procedure GRASP
    while (termination condition not met) do
        s ←ConstructGreedyRandomizedSolution
        ŝ ← LocalSearch(s)
        If f(ŝ) < f(s_best) then
            s_best ← ŝ
        end-if
    end-while
    return s_best
end-procedure
```

Figure 3.8 - High level pseudo-code for GRASP (Dorigo & Stützle, 2004)

Fogel & Michalewicz (2000) provide a GRASP application to solve a TSP with 70 cities. They randomly select a city to begin the tour and then add the other 69 cities one at a time to the tour. After constructing an initial solution, they run the algorithm and evaluate 2415 different solutions. In such big TSP problems, GRASP seems to find good solutions in reasonable amounts of time.

3.6 Particle Swarm Optimization (PSO)

PSO is inspired from the collective behaviors of animals. In this section, we will present a sample PSO algorithm to demonstrate how it works and talk about the kinds of problems it is applied to.

There are two key definitions in using PSO algorithms that have been defined in Section 2 earlier: position and velocity. The position and velocity of particle i at time t are represented by $x_i(t)$ and $v_i(t)$ respectively. The position and velocity of a particle changes based on the following equations:

$$\boldsymbol{x}_i(t) = \boldsymbol{x}_i(t-1) + \boldsymbol{v}_i(t-1) \tag{4}$$

equivalently, $\boldsymbol{x}_i(t)$ can be represented as a function of the previous position, previous velocity, $\boldsymbol{p}_i$, and $\boldsymbol{p}_g$ where, $\boldsymbol{p}_i$ is the local best position of particle i, and $\boldsymbol{p}_g$ is the neighborhood best position

$$\boldsymbol{x}_i(t) = f(\boldsymbol{x}_i(t-1), \boldsymbol{v}_i(t-1), \boldsymbol{p}_i, \boldsymbol{p}_g) \tag{5}$$

Equation (6) shows the velocity of particle i.

$$\boldsymbol{v}_i(t) = \boldsymbol{v}_i(t-1) + \Phi_1 (\boldsymbol{p}_i - \boldsymbol{x}_i(t-1)) + \Phi_2 (\boldsymbol{p}_g - \boldsymbol{x}_i(t-1)) \tag{6}$$

where;

Φ_1 and Φ_2 are randomly chosen parameters. Φ_1 represents the individual experience and Φ_2 represents the social communication. In figure 3.9 the PSO algorithm is given for n particles:

For i = 1 to n :
  If F(x_i) > F(p_i) then :
    For d = 1, . . . , D :
      $p_{id} = k_{id}$ // p_{id} is thus the best found individual
    end d
  end if
  $g = i$
  For j =index of the neighbors:
    If F(p_j) > F(p_g) then:
      $g = j$ // g is the best individual in the neighborhood
    end if
  end j
  For d = 1, . . . , D :
    $v_{id}(t) = v_{id}(t-1) + \Phi_1 (p_{id} - x_{id}(t-1)) + \Phi_2 (p_{gd} - x_{id}(t-1))$
    $v_{id} \in (-V_{max} + V_{max})$
    $v_{id}(t) = x_{id}(t-1) + v_{id}(t)$
  end d
end i
end

Figure 3.9 - The PSO algorithm for n particles (Dréo et al., 2006)

As seen in Figure 3.9, this algorithm can be used in multiple dimensions.
This PSO algorithm can be applied to many problems in the real life such as the TSP, the vehicle routing problem, the flowshop scheduling problem, etc. However, it is more commonly used in training of artificial neural networks.

4. Tabu search applications

TS applications comprise diverse fields like scheduling, computer channel balancing, cluster analysis, space planning, assignment, etc. It also has applications in many different technical problems like the travelling salesman, graph coloring, character recognition, etc. We have reviewed the recent literature (2000 and after) using keywords such as "tabu search", "local search", and "meta-heuristic." We collected around 150 articles most of which are focused on TS. In this chapter we will go over these articles to point out various applications of TS. Rather than going over all 150 articles, we only focused on the ones that represent diverse applications of TS.
Based on our literature review, TS is used widely on machine scheduling and job-shop scheduling problems. In his study Glover (1990), stated that Widmer & Hertz's (1990) application of TS to flow shop sequencing problems succeeded in obtaining solutions superior to the best previously found by applying a range of methods in about 90% of the cases. TS has shown superior results in other recent applications as well. Blazewicz et al. (2008) presented three meta-heuristics SA, TS, and variable neighborhood search (VNS) for the two-machine flow-shop problem with weighted late work criterion and common due date. Initial solutions were generated by Johnson's algorithm (1954) or list scheduling

algorithm which is a constructive method, that builds a solution by executing jobs selected according to a given priority dispatching rule. In order to have the best settings for the corresponding meta-heuristics, some parameters were tuned. As a result of the experiments TS control parameters were selected as: the neighborhood was generated by interchanging jobs based on the weighted processing times of the jobs, from 33% of generated jobs, termination condition was double the number of jobs, tabu list length was equal to 300% of the number of jobs. For comparing the efficiency of meta-heuristics, 20 problem instances were evaluated. Tuning improved TS performance by 7%. In this study, SA generated better schedules in shorter time. Chen et al. (2007) studied an extension of the hybrid flow-shop scheduling problem. In their study, a mixed-integer programming (MIP) model was provided, then a TS based algorithm was used. Also, Fink & Voss (2003) examined the application of different kinds of heuristic methods to the continuous flow-shop scheduling problem. Results show that effectiveness of heuristics depends on the problem size, desired solution quality, and available running time meaning that there is no single ''best'' method that dominates all other heuristics. In general, reactive TS (where tabu list is shortened or increased according to a resampling condition) obtained high-quality results without any parameter tuning. Brucker et al. (2003) presented a TS approach for the flowshop problem with intermediate buffers where different job sequences on the machines were allowed. Pan et al. (2008) proposed a PSO algorithm for a no-wait flowshop scheduling problem. When implementing the PSO algorithm, they also made comparisons of the performance of PSO with several other meta-heuristic procedures. One comparison was in between PSO and TS on this problem. It was mentioned that the TS algorithm and its hybrids generated better results on this specific problem. Eksioglu et.al. (2008) also used a TS algorithm utilizing changing neighborhoods for a flowshop scheduling problem. In their study, the results of this algorithm (3XTS) were compared to neuro-tabu search (EXTS) and ant colony optimization (ACO) algorithms. The property of the used TS algorithm in their research was that TS used three different neighborhood structures to obtain better results which meant it diversified the search. They observed that this TS algorithm obtained as good solutions as the neuro-TS and ACO algorithms. The computational time of 3XTS was almost the half of EXTS because it was capable to explore block properties. They concluded that the solution times of the ACO and the 3XTS algorithms were almost the same but 3XTS gived solutions which were closer to the optimal solution. Chen et al. (2008) developed a hybrid TS (HTS) for re-entrant permutation flow-shop scheduling problems. The proposed algorithm (HTS) improved the efficiency of TS obtaining favorable solutions within a reasonable time. It was mentioned that HTS found optimal solutions for all small problems. For large problems, HTS was superior to pure TS. Moreover, as the size of the problem increased, HTS performed better than pure TS.

Zhang et al. (2007) studied job-shop scheduling problems (JSP) proposing a TS algorithm with a new neighborhood structure that could avoid cycling and investigate much larger part of the solution space. Kis (2003) had also solved alternative JSP problems by TS and by a GA. Based on the results, TS was superior to GA both in terms of solution quality and computation time. Pezzella & Merelli (2000) presented a TS method guided by shifting bottleneck for the JSP. Liaw (2000) had developed a hybrid GA based on TS for open shop scheduling problem. It was stated that the algorithm performed extremely well Liaw (2003) examined the problem of scheduling two-machine preemptive open shops. A TS approach was proposed that provided excellent results. In most of the cases TS found optimum

values. In other cases it found values with average deviation from optimum value by nearly 2%. However, McMullen & Frazier (2000) dealt with the problem of mixed-model sequencing with multiple objectives on a just-in-time line. They used SA for solving the problem. It was stated that SA outperformed TS in most of the cases. Grabowski & Pempera (2000) dealt with sequencing of jobs in a production system presenting a TS a solution method. Vinicius et al. (2000) used TS for scheduling on identical parallel machines to minimize mean tardiness. Kim et al. (2003) also used a due date density-based categorizing heuristic that incorporated TS for parallel machines scheduling.

Waligóra (2008) studied discrete–continuous (discrete and continuous resources) project scheduling with discounted cash flows. Applications of TS, as well as simple search methods have been described. Based on the experiments, TS seemed to be an efficient algorithm for solving the considered problem, clearly outperforming simple search algorithms and producing results close to optimum. Mika et al. (2008) also studied a multi-mode resource-constrained project scheduling problem with schedule dependent setup times. TS was compared to a multi-start iterative improvement method and a random sampling method. Experiments showed that TS outperformed the other algorithms. Valls et al. (2003) dealt with the resource-constrained project scheduling problem through a non-standard implementation of TS principles.

Rubrico et al. (2008) studied scheduling of multiple picking agents for warehouse management. They developed a tabu scheduler exhibiting less computation time. Burke et al. (2004) proposed a TS hyperheuristic for timetabling and rostering. It outperformed GA in terms of feasibility while GA was better in terms of cost.

Burke & Smith (2000) developed a hybrid algorithm that was a memetic algorithm using TS for maintenance scheduling problem. It was concluded that this algorithm performed better at the expense of a little increase in solution time. Based on the results presented in the above references, we can conclude that a pure TS or hybrid TS performs superior to other advanced meta-heuristics like SA, GA or PSO and simple heuristics for flowshop scheduling, JSP and general scheduling problems.

Like scheduling, vehicle routing is another field in which TS is widely used. Teng et al. (2003), performed a comparative study of meta-heuristics for the vehicle routing problem (VRP) with stochastic demands. In this study, they present three meta-heuristics: SA, threshold accepting (TA), and TS meta-heuristic for the single VRP with stochastic demands. It is shown that quality of initial solutions has a positive effect on the TS algorithm while this is not valid for SA and TA. In this study, solution quality of TS outperforms the other two. Moreover, the superiority of TS increases as the problem size increases. Also TS requires less computation time than SA. Same neighborhood is used for all of the meta-heuristics. Hence, it is concluded that TS performs better than SA and TA in terms of solution quality and computation time for the single VRP with stochastic demands. Fallahi et al. (2008) introduced a multi-compartment vehicle routing problem (MC-VRP), a problem not yet studied in literature in spite of its important practical applications. Three algorithms were proposed to solve it: a constructed heuristic, a memetic algorithm (MA) combined with a path relinking method used as post optimization, and TS. These methods have been tested using two sets of problems. It is stated that in general, TS provides slightly better solutions than MA but requires more computational time. Li et al. (2007) compared the results of 11 algorithms that solve the open vehicle routing problem (OVRP) and found that procedures based on adaptive large neighborhood search, record-to-record travel, and TS performed

well. Shetty et al. (2008) considered the strategic routing of a fleet of unmanned combat aerial vehicles (UCAVs). The TS heuristic for TSP calculates quick tours when the assignments are finalized. For larger scale problems the TS heuristic provides good feasible solutions quickly. Iterative nature of the heuristic allows it to be stopped after a feasible solution is generated. It is stated that such problems might benefit from the decomposition scheme proposed in this work with TS coordinating the subproblems. Caricato et al. (2003) have examined a VRP under Track Contention whose applications arise in automated material handling systems and flexible manufacturing systems. Three heuristics were developed: two simple procedures and a tailored TS. In view of its adaptation to a real-time setting, the TS algorithm was parallelized. Based on the results, TS provides better solutions although this usually comes at the expense of a larger computing time. Backer et al. (2000) dealt with VRPs presenting a method for combining constraint programming with local search including TS. Gendreau et al. (2007) proposed a TS heuristic for the vehicle routing problem with two-dimensional loading constraints.

TS is used in facility location problems as well. Arostegui et al. (2006) compared relative performance of TS, SA, and GA on various facility location problems (FLP). In their study it is mentioned that another comparison between these meta-heuristics was made in 1993 by Sinclair for the quadratic assignment problem (QAP). In the FLP problem GA performed worst and TS provided better solutions than SA in 28 instances out of 37 having nearly the same computation time. In 1992 Kincaid compared TS and SA for noxious facilities location and concluded that TS outperformed SA. In 2003, Wang et al. showed that for the budget constraint location problem TS results were more satisfactory in terms of quality when compared to a Lagrangian Relaxation (LR) approximation. Also Chamberland in 2004 showed that for a network subsystem expansion problem a TS based heuristic provided good solutions. Capacitated (CFLP), multi-period (MPFLP), and multi-commodity (MCFLP) facility location problems were used to compare three meta-heuristics. Their performance was evaluated on three dimensions: computation time limitation, solution limited dimension, unrestricted dimension. For time-limited results, performance of TS was superior to others in CFLP and MPFLP. Statistically TS showed best performance for rapidly reaching low-cost solutions followed by SA and GA. For MCFLP, GA provided best result. Vector representation for MCFLP resulted in fewer solutions to be evaluated. Overall, given the same amount of time TS in general gave the best results. For solutions-limited results, SA performed similar to or better than TS. For unrestricted results, TS performed best for CFLP and MCFLP while SA performed best for MPFLP. Michel & Hentenryck (2004) presented a simple tabu-search algorithm which performed well for un-capacitated warehouse location problem (UWLP). It outperformed the GA in efficiency and robustness. Amaldi et al. (2006) investigated mathematical programming models for base station location and configuration. They proposed a TS algorithm starting with an initial solution using a randomized greedy procedure. Mladenovic et. al (2003) proposed a TS algorithm and variable neighborhood search (VNS) for *p*-Center problem, one of the basic models in discrete location theory. Cortinhal & Captivo (2003) dealt with the single source capacitated location problem. They proposed a Lagrangean heuristic combined with TS.

The Capacitated Arc Routing Problem (CARP) is another application for TS. Brandão and Eglese (2008) presented a deterministic TS algorithm for the CARP. Results in this study show that TS is capable of providing high quality results in a reasonable computing time. Amberg et al. (2000) dealt with multiple center CARP with a TS algorithm using capacitated

trees. They concluded that the proposed TS is capable of yielding good solutions. Brandao and Eglese (2006) solved a capacitated arc routing problem using a deterministic TS algorithm and they compared the results with a memetic algorithm and CARPET (their heuristic for solving CARP). They concluded that the results of the TS algorithm were reproducible because it was a deterministic algorithm. On the other hand, they denoted that the results of the memetic algorithm and CARPET could not be reproduced because these algorithms included random elements. Since the memetic algorithm and CARPET used random elements, everytime they run the algorithms they obtained different results so they preferred the results of the TS algorithm.

In the network design field, Ignacio et al. (2008) used TS in a computer network design problem called the concentrator location problem. A problem formulation and a LR procedure were presented. Approximate solutions are obtaind by TS. Computational results are shown for 320 problem instances. For problems with more than 100 users TS provides better results than CPLEX. Two other studies in network design were by Chamberland (2003) and Fortz et al. (2003).

In production and distribution field, Russell et al. (2008) studied a problem for integrating the production and distribution of newspapers from plant to bulk delivery locations. The proposed TS methodology made it easier and more efficient for the newspaper to handle increasing volume of pre-print advertisement. TS methodology was able to reduce the number of vehicles required by 18.18%. Lukac et. al. (2008) studied production planning problems with sequence dependent setups and solved them by a TS based heuristic. Valdes et al. (2007) studied a two-dimensional non-guillotine cutting problem proposing a TS algorithm. Based on the computational results, TS worked well for the constrained and double-constrained test problems. Onwubolu et al. (2000) proposed a TS approach to cellular manufacturing systems. In this study a cell formation problem was modeled that had three objectives: minimization of intercellular movements, minimization of cell load variation, and a combination of two. It was stated that TS worked as well as other published algorithms for the same problem. TS has an extra advantage that is allowing the designer to select the maximum number of cells as well as machines in a cell. Hung et al. (2003) used TS with ranking candidate list to solve production planning problems with setups. Pai et al. (2003) dealt with optimization of laminate stacking sequence for failure load maximization using TS. Results were comparable to GA.

There are many other application fields and problems in which TS is used. For example: Cell planning with capacity expansion in mobile communications (Lee & Kang, 2000), application-level synthesis methodology for multidimensional embedded processing systems (Alippi et al. 2003). Cogotti et al. (2000) performed a comparison of optimization techniques for Loney's Solenoids Design and proposed an alternative TS algorithm. Emmert et al. (2003) have shown an effective way of bi-partitioning electrical circuits using TS. It was stated that TS offered quick convergence to good partitioning solutions for circuits in the range of their application. Their algorithms show dramatic improvement in execution time with good solution quality as compared to a random move SA approach. They also mention that their placement method is suitable for quickly initializing the inputs to other non-deterministic placement algorithms. Rajan et al. (2003) proposed a neural-based TS method for solving unit commitment problem. Blazewicz et al. (2000) proposed a TS-based algorithm for DNA sequencing in the presence of false negatives and false positives. Corberan et al. (2000) studied a mixed rural postman problem in which TS was used. Ahr &

Reinelt (2005) presented a TS algorithm for the min-max Chinese postman problem. Tan et al. (2008) proposed an optimization procedure combining zonation methods with TS to identify the spatial distribution of a hydraulic conductivity field. Blöchliger & Zufferey (2008) studied a graph-coloring problem proposing a heuristic using partial solutions and a reactive tabu scheme. It was shown that this reactive tabu scheme obtains good results on a large sample of benchmark graphs which are generally difficult to color. Konak et al. (2003) studied a reliability design problem called the Redundancy Allocation Problem. A TS, named TSRAP,was described and compared to other approaches to the problem.
As amply demonstrated, TS is applied to a large variety of problems that arise in different fields. Next, we discuss performance of TS compared to other meta-heuristics.

5. Performance of tabu search

In 1988, the Committee on the Next Decade of Operations Research (CONDOR) evaluated TS together with SA and GA to be "extremely promising" for the future treatment of practical applications (Glover, Laguna, 1997). Based on section 4, we can state that the committee predicted the future quite well. TS, incorporating many artificial intelligence properties, outperformed other meta-heuristics in many application fields. However, theoretical aspects of TS that make it successful are still a matter of discussion. Some nice properties of TS are: TS generally proceeds more aggressively to local optimum unlike SA which relies on the premise that a slow descent will lead to a local optimum that is closer to a global one. This rationale of TS derives from two considerations: (1) optimization problems can be solved making the best available move at each iteration; (2) rather than spending more time in regions whose solutions are less attractive TS devotes larger effort to exploring regions where solutions are good (Glover, 1989). SA may not be stopped at any desired moment in time since the control parameter (temperature) has to converge to a value close to zero to obtain a meaningful implementation, the cooling schedule needs to be tuned to the time available for deriving a solution (Michiels et al., 2007). TS can be stopped at any time. Iterative nature of the TS may allow it to be stopped after a feasible solution is found. GA is a population based approach. It requires evaluation of populations over generations. For complex problems that kind of approach results in a great computational effort. TS, considering neighboring moves and not needing objective function gradient information as GA, is more efficient as demonstrated by Konak et al. (2003). SA makes stochastic moves. However TS uses deterministic moves which reduces variability due to initial solutions and other parameters.
However, it is obvious that for a successful implementation of TS it is necessary to tune the algorithm for the specific problem on hand. Jaeggi et al. (2008) states that "A major shortcoming in the assessment of optimization algorithms for use on real-world problems is the lack of a suitable set of benchmark problems, which accurately capture the main features of the problems of interest. Until such a set is developed, true performance comparison between algorithms is difficult and one must rely on inference from a less suitable set of benchmark problems." Based on the experiments discussed in our review of the literature, tabu lists designed to prevent repetition rather than reversal of moves seem to not work well. An empirical discovery for the application of TS methods is that the number of iterations has a highly stable range of values that prevents both cycling and leads to good solutions (Glover, 1989). Development of TS is an iterative process, thus hoping to find the best solution at the very beginning would be naive. Necessary modifications depending on the problem instance must be made.

As mentioned in (Dréo et al. 2006) for many applications TS based heuristics showed more effective results. TS owes its efficiency to rather fine tuning of a large collection of parameters which may seem counter-intuitive. TS is a trajectory method constituting a neighborhood of solutions at each iterations in contrast to SA generating a single solution. In some cases, that attribute of TS makes it slower than SA as shown by Blazewicz et al. (2008). Deterministic TS may be feasible to be able to reproduce the results as shown by Brandão & Eglese (2008). GA appears to perform well in an environment when information is limited as in MCFLP. It seems like the longer the solution time the better the probability that TS will show superior performance. Local-search component and constraint handling flexibility of TS makes it attractive for problems having many constraints (Jaeggi et al., 2008). Vallada et al. (2008) stated that from the meta-heuristics tested, the two SA algorithms proposed outperform all other methods evaluated. They have also shown that the TS methods are good meta-heuristics in the m-machine flowshop problem with the objective of minimizing total tardiness. The method used for intensification by TS or GA, plays an important role in determining the accuracy of the results. Liaw (2000) stated that GA is good at performing global search and TS is effective for fine tuning for the open shop scheduling problem. A hybrid approach combining GA and TS in that study found optimal solution for nearly all test problems. Incorporation of appropriate heuristics with pure TS may be more effective as shown by Chen, et al. (2008). If the evaluation of the entire neighborhood space requires too much computation, then a neighborhood sorting method as well as using a ranking candidate list strategy may improve the performance of TS (Hung et al., 2003).

6. Conclusion

In today's global and competitive environment, the use of scarce resources in the best possible way is more important than ever, and time is one of the critical resources for almost all problems. In this book chapter, we tried to show how efficient and effective results can be obtained using meta-heuristic methods, specifically TS.

According to editorial of European Journal of Operational Research (EJOR) (Editorial, 2007), meta-heuristic research efforts seem to be aimed at two main areas of application: production/scheduling problems and logistics problems. Other domains of applications in a related issue of EJOR are finance, product design, bio-computing and data mining. A common feature seen is that authors use hybrid meta-heuristics to acquire efficient tailor-made solution approaches.

Hybrid approaches seem to have better performance in some problems motivating researchers to focus on hybrid meta-heuristic usage. For example, Ting et al. (2003) presented a work focusing on a novel mating strategy, called tabu genetic algorithm (TGA). TGA integrates tabu search (TS) into GA's selection. Structures of GA and TS are not modified in these approaches. It is shown that in contrast to running GA and TS alternately, TGA implants characteristics of TS like aspiration into GA's mating strategy. It is concluded that TGA outperforms GA, TS, and conventional hybrids of GA and TS, in terms of solution quality and convergence speed.

Glover (2007) mentions that from the beginning of TS journey to present enabled us to solve many kinds of optimization problems effectively although there is a long way to go. Glover suggests focusing on human memory usage to understand how it affects problem solving, so that these features can be incorporated to TS memory. For this case psychology and heuristics fields should engage in new projects.

7. References

Aarts, E. & Lenstra, J., K. (2003). *Local Search in Combinatorial Optimization*, Princeton University Press, 0-691-11522-2, New Jersey

Ahr, D. & Reinelt G. (2005). A tabu search algorithm for the min–max k-Chinese postman problem, *Computers and Operations Research*, 33 (7 December 2005), 3403-3422, 0305-0548.

Alippi, C., Galbusera A., & Stellini M. (2003). An Application-Level Synthesis Methodology for Multidimensional Embedded Processing Systems, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 22, No. 11, 0278-0070

Alves, M. J. & Climaco, J. (2000). an Interactive Method for 0-1 Multiobjective Problems Using Simulated Annealing and Tabu Search, *Journal of Heuristics*, 6, 2000, 385–403

Amaldi, E.; Belotti, P.; Capone, A. & Malucelli, F. (2006). Optimizing Base Station Location and Configuration, *Ann Oper Res*, 146, 27 June 2006, 135–151

Amberg, A.; Domschke, W. & Voss S. (1999). Multiple Center Capacitated Arc Routing Problems: A Tabu Search Algorithm Using Capacitated Trees, *European Journal of Operational Research,* 124, 2000, 360-376, 0377-2217

Armentano, V.A. & Yamashita D.S. (2000). Tabu search for scheduling on identical parallel machines to minimize mean tardiness, *Journal of Intelligent Manufacturing*, 453-460 Arostegui, M., A.; Kadipasaoglu, S., N. & Khumawalab, B., M. (2006), an Empirical Comparison of Tabu Search, Simulated Annealing, and Genetic Algorithms for Facilities Location Problems. *Int. J. Production Economics*, 103, 2006, 742–754, 0925-5273

Backer, B.E. & Furnon, V. & Shaw, P. (2000). Solving Vehicle Routing Problems Using Constraint Programming and Meta-heuristics, *Journal of Heuristics,* 501-523

Blazewicz, J.; Formanowicz, P.; Kasprzak, M.; Markiewicz, W., T. & Weßglarz, J. Tabu Search for DNA Sequencing with False Negatives and False Positives, *European Journal of Operational Research,* 125, 2000, 257-265, 0377-2217

Blazewicza, J.; Peschb,E.; Sternaa,M. & Wernerc F. (2006). Meta-heuristic Approaches for the Two-Machine Flow-Shop Problem with Weighted Late Work Criterion and Common Due Date. *Computers & Operations Research,* 35, 2008, 574 – 599, 0305-0548

Blöchligera, I. & Zuffereyb, N. (2006). a Graph Coloring Heuristic Using Partial Solutions and a Reactive Tabu Scheme, Computers & Operations Research, 35, 2008, 960 – 975, 0305-0548

Blum, C. & Roli, A. (2003). Meta-heuristics in Combinatorial Optimization: Overview and Conceptual Comparison, *ACM Computing Surveys,* Vol. 35, No. 3, September 2003, 268–308Brandão, H. & Eglese, R. (2006). A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers and Operations Research,* 35 (25 September 2006),1112-1126, 0305-0548

Brandãoa, J. & Egleseb R. (2006). a Deterministic Tabu Search Algorithm for the Capacitated Arc Routing Problem. *Computers & Operations Research,* 35, 2008, 1112 – 1126, 0305-0548

Brucker, P.; Heitmann, S. & Hurink, J. (2003). Flow-shop problems with intermediate buffers, *OR Spectrum,* 25, 2003, 549–574

Budenbender, K.; Grunert, T. & Sebastian, H. (2000). a Hybrid Tabu Search/Branch-and-Bound Algorithm for the Direct Flight Network Design Problem, *Transportation Science,* 34, 4, November 2000, 364–380, 1526-5447

Burke, E., K. & Smith A. J. (2000). Hybrid Evolutionary Techniques for the Maintenance Scheduling Problem, *IEEE Transactions On Power Systems*, 15, 1, February 2000, 122-128, 0885-8950

Burke, E.K., Kendall, G. & Soubeiga, E. (2004). A Tabu-Search Hyperheuristic for Timetabling and Rostering, *Journal of Heuristics,* August 2003,451-470

Caricato, P.; Ghiani, G.; Grieco, A. & Guerriero, E. (2002). Parallel Tabu Search for a Pickup and Delivery Problem under Track Contention, *Parallel Computing,* 29, 2003, 631–639, 0167-8191

Chamberland, S. (2003). on the Overlay Network Design Problem for the Soft-Label Switched Paths in IP Networks, *INFOR,* 41, 4, Nov. 2003, 301-332

Chelouah, R. & Siarry, P. (2000). Tabu Search Applied to Global Optimization. *European Journal of Operational Research,* 123, 2000, 256-270, 0377-2217

Chen, J.; Pan, J., C. & Wu, C. (2008). Hybrid Tabu Search for Re-Entrant Permutation Flow-Shop Scheduling Problem, *Expert Systems with Applications,* 34, 2008, 1924–1930, 0957-4174

Chen, Lu.; Bostel, N.; Dejax P.; Cai J. & Xi L. (2006). a Tabu Search Algorithm for the Integrated Scheduling Problem of Container Handling Systems in a Maritime Terminal, *European Journal of Operational Research*, 181, 2007, 40–58, 0377-2217

Cogotti, E.; Fanni, A. & Pilo, F. (2000). A Comparison of Optimization Techniques for Loney's Solenoids Design: An Alternative Tabu Search Algorithm, *IEEE Transactions On Magnetics,* 36, 4, July 2000, 1153-1157, 0018–9464

Corberan, A.; Marti, R. & Romero, A. (1998). Heuristics for the Mixed Rural Postman Problem, *Computers & Operations Research*, 27, 2000, 183-203, 0305-0548

Cortinhal, M.J. & Captivo, M.E. (2003). Upper and lower bounds for the single source capacitated location problem, *European Journal of Operations Research,* 333-351, 0377-2217

Dorigo, M. & Stützle, T. (2004). Ant Colony Optimization, *The MIT Press,* ISBN 0-262-04219-3, United States of America

Dréo, J.; P´etrowski, A.; Siarry, P. & Taillard, E. (2006). *Meta-heuristics for Hard Optimization,* Springer-Verlag Berlin Heidelberg, 10 3-540-23022, New York

Editorial, (2007). Applications of Meta-heuristics. *European Journal of Operational Research*, 179, 2007, 601–604, 0377-2217

Eksioglu, B., Eksioglu, S. D. and Jain, P. (2008). A tabu search algorithm for the flowshop scheduling problem with changing neighborhoods, *Computers and Industrial Engineering,* 54 (13 April 2007),1-11, 0360-8352

Emmert, J., M.; Lodha, S. & Bhatia, D., K. (2003). On Using Tabu Search for Design Automation of VLSI Systems, *Journal of Heuristics*, 9, 2003, 75–90

Fallahi, A.; Prins, C. & Calvo, R., W. (2006). a Memetic Algorithm and a Tabu Search for the Multi-Compartment Vehicle Routing Problem, *Computers & Operations Research*, 35, 2008, 1725 – 1741, 0305-0548

Fink, A. & Voß S. Solving the Continuous Flow-Shop Scheduling Problem by Meta-heuristics, *European Journal of Operational Research*, 151, 2003, 400–414, 0377-2217

Fortz, B.; Soriano, P. & Wynants, C. (2003). a Tabu Search Algorithm for Self-Healing Ring Network Design, *European Journal of Operational Research*, 151, 2003, 280–295, 0377-2217

Gendreau, M., Lori, M., Laporte, G. & Martello, S. (2007). A Tabu Search Heuristic for the Vehicle Routing Problem with Two-Dimensional Loading Constraints, *Interscience*, Vol. 51(1), October 2005, 4-18

Glover, F. & Laguna, M. (1997). Tabu Search, *Kluwer Academic Publishers*, 0-7923-8187-4, Massachusetts

Glover, F. (1988). Tabu Search – Part I, *ORSA Journal on Computing*, Vol.1, No.3, Summer 1989, 0899-1499/89/0103-0190

Glover, F. (1989). Tabu Search – Part II, *ORSA Journal on Computing*, Vol.2, No.1, Winter 1990, 0899-1499/90/0201-0004

Glover, F. (2006). Tabu search—Uncharted Domains, *Ann. Oper. Res.* 149, 2007, 89–98

Grabowski, J. & Pempera, J. (2000). Sequencing of jobs in some production system, *European Journal of Operations Research,* 1 March 1999,535-550, 0377-2217

Hasan, M.; AlKhamis, T. & Ali J. (2000). a Comparison between Simulated Annealing, Genetic Algorithm and Tabu Search Methods for the Unconstrained Quadratic Pseudo-Boolean function, *Computers & Industrial Engineering*, 38, 2000, 323-340, 0360-8352

Hertz, A. & Werra, D. (1990). The Tabu Search Meta-heuristic: How We Used It, *Annals of Mathematics and Artificial Intelligence,* 1, 1-4, September (1990), 111-121, 1573-7470

Hindsberger, M. & Vidal, R. (2000). Tabu Search – A Guided Tour, *Control and Cynernetics*, 29, 3, 2000

Holder, A. editor. Mathematical Programming Glossary. *INFORMS Computing Society*,
http://glossary.computing.society.informs.org/, 2006-2007. Originally authored by Harvey J. Greenberg, 1999-2006.

Hung, Y.F., Chen, C.P., Shih, C.C. & Hung, M.H. (2003). Using tabu search with ranking candidate list to solve production planning problems with setups, *Computers & Industrial Engineering,* 11 September 2003, 615-634), 0360 8352

Ignacio, A., A., V.; Filho, V., J., M., F. & Galvão R., D. (2006). Lower and Upper Bounds for a Two-Level Hierarchical Location Problem in Computer Networks, *Computers & Operations Research*, 35, 2008, 1982 – 1998, 0305-0548

Imahori, S., Yagiura, M., & Ibaraki T. (2003). Local search algorithms for the rectangle packing problem with general spatial costs, *Springer,* November 25 2002, 543-569

Jaeggi, D. M., Parks, G.T., Kipouros, T. & Clarkson P.J. (2006), The development of a multi-objective Tabu Search algorithm for continuous optimisation problems. *European Journal of Operational Research,* 185 (25 October 2006),1192-1212, 0377-2217

Jaeggi, D., M.; Parks, G., T.; Kipouros, T. & Clarkson P.J. (2006). the Development of a Multi-Objective Tabu Search Algorithm for Continuous Optimisation Problems, *European Journal of Operational Research*, 185, 2008, 1192–1212, 0377-2217

Jiang, T.; Luo, A.; Li, X. & Kruggel, F. (2003). a Comparative Study of Global Optimization Approaches to Meg Source Localization, *Intern. J. Computer Math.,* 80(3), 2003, 305–324, 1029-0265

Johnson, S. (1954). Optimal Two and Three Stage Production Schedules with Set-up Times Included, *Naval Research Logistics Quarterly,* 1, 1, 1954, 61-68

Kidwai, F.A., Marwah B.R., Deb, K. & Karim M. R. (2005). A Genetic Algorithm Based Bus Scheduling Model for Transit Network, *Proceedings of the Eastern Asia Society for Transportation Studies,* Vol. 5,477 – 489

Kim, S.S., Shin H.J., Eom, D.H. & Kim, C.O. (2003). A due date density-based categorising heuristic for parallel machines scheduling, *International Journal of Advanced Manufacturing Technology,* 26 July 2003,753-760

Kis, T. (2003)Job-shop scheduling with processing alternatives European Journal of Operational Research, 151, 2003, 307–332, 0377-2217

Konak, S.;Smith, A. & Coit, D. (2003). Efficiently Solving the Redundancy Allocation Problem Using Tabu Search, *IIE Transactions*, 35, 515-526, 2003

Laurent, M. & Hentenryck, P., V. (2004). a Simple Tabu Search for Warehouse Location European Journal of Operational Research, 157, 2004, 576–591, 0377-2217

Lee, C., Y. & Kang, H., G. (2000). Cell Planning with Capacity Expansion in Mobile Communications: A Tabu Search Approach, *IEEE Transactions On Vehicular Technology,* 49, 5, September 2000, 1678-1691, 0018–9545

Lia, F.; Goldenb, B. & Wasilc, E. (2006). the Open Vehicle Routing Problem: Algorithms, Large-Scale Test Problems, and Computational Results, *Computers & Operations Research,* 34, 2007, 2918 – 2930, 0305-0548

Liaw, C. (1999). a Hybrid Genetic Algorithm for the Open Shop Scheduling Problem, *European Journal of Operational Research,* 124, 2000, 28-42, 0377-2217

Liaw, C.F. (2003). An efficient tabu search approach for the two-machine preemptive open shop scheduling problem, *Computers and Operations Research,* 1 September 2002, 2081-2095, 0305-0548

Lukac, Z., Soric, K. & Rosenzweig, V., V. (2006). Production Planning Problem with Sequence Dependent Setups as a Bilevel Programming Problem, *European Journal of Operational Research,* 187, 2008, 1504–1512, 0377-2217

McMullen, P.R. & Frazier, G.V. (2000). A simulated annealing approach to mixed-model sequencing with multiple objectives on a just-in-time line, 01 August 2000, *IIE Transactions,* 679-686

Merz, P. & Freisleben, B. (2000). Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem, *IEEE Transactions on Evolutionary Computation,* Vol. 4, No. 4 (November, 2000) 16 pages (337-352), 1089–778X

Michalewicz, Z. & Fogel, D.B. (2000). *How to Solve it : Modern Heuristics*, Springer, ISBN 3-540-22494-7, Germany.

Michiels W.; Aarts, E. & Korst, J. (2007). *Theoretical Aspects of Local Search*, Springer-Verlag Berlin Heidelberg, 3-540-35853-6, New York

Mika M., Waligóra, G.& Węglarz J. (2006). Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times. *European Journal of Operational Research,* 187 (15 November 2006), 1238-1250, 0377-2217

Mika, M.; Waligora G. & Weglarz, J. (2005). Tabu Search for Multi-Mode Resource-Constrained Project Scheduling with Schedule-Dependent Setup Times, *European Journal of Operational Research,* 187, 2008, 1238–1250, 0377-2217

Mladenović, N., Labbé, M. & Hansen, P. (2003). Solving the p-Center Problem with Tabu Search and Variable Neighborhood Search, *Networks,* Vol. 42, No. 1, 48-64

Mladenović, N., Petrović, J., Kovačević-Vujčić, V. & Ćangalović, M. (2003). Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search, *European Journal of Operations Research,*389-399, 0377-2217

Onwubolu, G. & Songore, V. (2000). a Tabu Search Approach to Cellular Manufacturing Systems, *Production Planning & Control*, 11, 2, 2000, 153–164, 0953-7287

Pai, N., Kaw, A. & Weng, M. (2003). Optimization of laminate stacking sequence for failure load maximization using Tabu search, *Composites,* 27 August 2002,405-413, 1359-8368

Palubeckis, K. (2004), Multistart Tabu Search Strategies for the Unconstrained Binary Quadratic Optimization Problem. *Annals of Operations Research,*259-282

Pan, Q. K., Tasgetiren, M.F. & Liang, Y.C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem, *Computers & Operations Research,*2807-2839, 0305-0548

Perez, J.; Moreno-Vega, J., M. & Martin, I., R. (2003). Variable Neighborhood Tabu Search and Its Application to the Median Cycle Problem, European Journal of Operational Research, 151, 2003, 365–378, 0377-2217

Pezzella, F. & Merelli E. (2000). A tabu search method guided by shifting bottleneck for the job shop scheduling problem, *European Journal of Operations Research,* 1 September 1998,297-310, 0377-2217

Rajan, C.C.A., Mohan M.R. & Manivannan, K. (2003). Neural-based tabu search method for solving unit commitment problem, *IEE Proceedings Online,* Vol. 150, No. 4, July 2003,469-475

Rubrico, F.I.U., Ota, F., Higashi, T. & Tamura, H. (2008). Meta-heuristic scheduling of multiple picking agents for warehouse management, *Industrial Robot: An International Journal,*58-68, 0143-991X

Russella, R.; Chianga, W. & Zepedab, D. (2006). Integrating Multi-Product Production and Distribution in Newspaper Logistics, *Computers & Operations Research*, 35, 2008, 1576 – 1588, 0305-0548

Shetty, V., K.; Sudit, M. & Nagi R. (2006). Priority-Based Assignment and Routing of a Fleet of Unmanned Combat Aerial Vehicles, *Computers & Operations Research*, 35, 2008, 1813 – 1828, 0305-0548

Tan, C.C., Tung, C.P. & Tsai, F.T.C. (2008). Applying zonation methods and tabu search to improve the ground-water modeling, *Journal of the American Water Resources Association,* May 2007,107-120

Teh, Y., S.; Rangaiah, G., P. Tabu Search for Global Optimization of Continuous Functions with Application to Phase Equilibrium Calculations, *Computers and Chemical Engineering,* 27, 2003, 1665-1679, 0098-1354

Teng, S.; Ong, H., L. & Huang, H., C. (2003). a Comparative Study of Meta-heuristics for Vehicle Routing Problem with Stochastic Demands, *Asia-Pacific Journal of Operational Research*, 20, 2003, 103-119

Ting, C.; Li, S. & Lee, C. (2002), On the Harmonious Mating Strategy through Tabu Search, *Information Sciences*, 156, 2003, 189–214, 0020-0255

Valdes, R., A., Parreno, F. & Tamarit, J., M. (2006). a Tabu Search Algorithm for a Two Dimensional Non-Guillotine Cutting Problem, *European Journal of Operational Research*, 183, 2007, 1167–1182, 0377-2217

Vallada E.; Ruiz, R. & Minella, G. (2006). Minimising Total Tardiness in the M-Machine Flowshop Problem: A Review and Evaluation of Heuristics and Meta-heuristics, *Computers & Operations Research*, 35, 2008, 1350 – 1373, 0305-0548

Valls, V., Quintanilla, S. & Ballestín F. (2003). Resource-constrained project scheduling: A critical activity reordering heuristic, *European Journal of Operations Research,* 282-301, 0377-2217

Waligóra, G. (2006). Discrete–Continuous Project Scheduling with Discounted Cash Flows—A Tabu Search Approach, *Computers & Operations Research*, 35, 2008, 2141 – 2153, 0305-0548

Waligóra, G. (2008). Discrete–continuous project scheduling with discounted cash flows—A tabu search approach, *Computers & Industrial Engineering,* 27 October 2006, 2141-2153, 0305-0548

Watson, J.; Beck, J., C.; Howe, A. & Whitley, L. D. (2002). Problem Difficulty for Tabu Search in Job-Shop Scheduling, *Artificial Intelligence*, 143, 2003, 189–217, 0004-3702

Zhang, C., Y.; Li, P.; Guan, Z. & Rao, Y. (2006). a Tabu Search Algorithm with a New Neighborhood Structure for the Job Shop Scheduling Problem, *Computers & Operations Research*, 34, 2007, 3229 – 3242, 0305-0548

2

A Multiobjective Tabu Framework for the Optimization and Evaluation of Wireless Systems

Katia Jaffrès-Runser[1,2], Jean-Marie Gorce[1] and Cristina Comaniciu[2]
[1] *CITI Laboratory, INSA Lyon*
[2] *Stevens Institute of Technology, Hoboken, NJ*
[1]*France*
[2]*United States of America*

1. Introduction

This chapter provides an insight on the use of Tabu search for the resolution of multiobjective (MO) optimization problems. Many real-world engineering problems are not defined with a single objective. Their complexity and the user demands lead to the use of multiple performance criteria that have to be optimized concurrently. For instance, when optimizing the shape of aerodynamic systems (Jaeggi et al., 2008) or when looking for the optimal placement of base stations in cellular networks (Reininger & Caminada, 2001), several optimization functions are necessary to account for the different performance metrics or services rendered by the system. In contrast to mono-objective optimization problems, there is rarely a unique solution that optimizes all the criteria. This is due to the fact that the objectives are often in conflict and that several feasible trade-off solutions exist, representing the best available feasible configurations of the system. Depending on the goal of the designer, one or several of these trade-off solutions can be eventually applied to the real implementation. This set of trade-off solutions is known as the Pareto-optimal set or as the optimal Pareto front.

The role of a MO optimization algorithm is to find the best possible representation of the Pareto front. Due to the complexity of the evaluation functions in real-world systems, and due to the high number of continuous or discrete variables that usually characterizes these optimizations, an exact resolution of the problem is rarely affordable within a reasonable computation time. Therefore, most of the search algorithms are based on well-known metaheuristics such as genetic algorithms, simulated annealing or local search techniques.

Main efforts in developing MO algorithms have been devoted to adapt genetic algorithms and evolution strategies to multiobjective optimization. The first multiobjective genetic algorithm was developed in 1985 and research in this field has been very active since (Deb, 2001). Simulated annealing techniques have also often been proposed. However, few MO search metaheuristics rely on Tabu search. A 2002 survey on multiobjective optimization techniques (Jones et al., 2002) showed at that time that Tabu search inspired techniques only represented 6% of the literature investigated (the survey was conducted over 115 articles), while 70% of the articles proposed a genetic / evolution-based strategy and 24% proposed a

simulated annealing implementation. However, Tabu Search (TS) gained more attention in the last years as several seminal works on MO Tabu Search have been published.
The purpose of this chapter is to provide the reader with a clear view on the use of TS in the field of multiobjective optimization and to outline its performance for real-world optimization problems. After a review of the main Tabu inspired MO optimization algorithms, a simple MO Tabu Search procedure developed by the authors (referred to as PMOTS) is introduced. This heuristic has been developed to resolve the wireless LAN access point planning problem (WLP problem). The WLP planning problem is used throughout the whole chapter as an illustrative example for the statements presented herein. Another real-world optimization problem in the field of wireless networks is also presented at the end of the chapter to provide another implementation example. This new example deals with the problem of performance evaluation for routing in a wireless sensor network, where routing performance benchmarks can be provided by the search for the Pareto optimal data forwarding patterns in the network.
The outline of this chapter is the following: Section 2 introduces the WLP problem as a multiobjective optimization problem. Section 3 presents the main concepts of multiobjective optimization and gives an overview of the main MO strategies developed so far. Section 4 concentrates on the use of Tabu Search for multiobjective optimization. It first presents the Tabu based MO metaheuristics found in the literature and then focuses on the description of the proposed MO-Tabu heuristic. In section 5, a discussion on the adaptation of MO-Tabu for the WLP resolution is given and the related results are presented in a first subsection. Then, the use of PMOTS for the evaluation of wireless sensor networks is discussed. Section 6 provides concluding remarks concerning the use of Tabu in a multiobjective optimization context.

2. MO optimization for wireless systems

2.1 The Wireless LAN Planning problem (WLP problem)

In the last decade, wireless LANs have experienced great success as lots of networks have been deployed in companies or private areas either as hot spots for public access or as private networks. More and more mobility-related applications such as Voice over IP for WiFi networks are being implemented on WLANs. The locations of the access points (AP) in the service area are key factors for design and strongly influence the performance of the network. For small networks, simple rules of thumbs (e.g. site surveys, quick installations, on-site network tuning) can be applied to plan the network. However, when the network grows (i.e. more than 10 APs) and a higher quality of service is required at the application level, automatic planning tools are needed to tackle such a problem.
In WLAN systems, bad performance originates from three main reasons: a lack of radio coverage in the design area (i.e. low Signal to Noise Ratio (SNR)), inter-cell interference (i.e. low Signal to Interference and Noise Ratio (SINR)) and a high number of users sharing the same AP. WLAN planning thus aims at coping with all of these issues by looking for the network configuration that minimizes one or more evaluation criteria. In the literature, early works mostly dealt with coverage constraints (Sherali et al., 1996). Interference being a strong limiting factor for wireless networks, interference mitigation constraints has then been introduced in the problem formulation (Aguado-Agelet, 2002). Finally, throughput considerations are also accounted for in the problem definition to ensure a substantial quality of service in the network (Bahri & Chamberland, 2005).

The WLP problem is clearly a multiobjective optimization problem. Most of the times, it has been formulated as a discrete minimization problem since the set of available locations for the APs is finite as it depends on geometry of the deployment area.

2.2 Formal definition of the WLP problem

The following combinatorial WLP problem definition has been proposed by the authors in (Jaffrès-Runser et al., 2008). A simplified description of the problem is provided herein. See (Jaffrès-Runser et al., 2008) for a more detailed description.

Let us define a discrete set of M candidate AP locations in a building floor (cf. Figure 1). Variables of the planning problem are the number N of APs to plan, their locations, their transmission powers and their directions of emission if the antennas are directional. The number of APs to plan, N, can either be fixed or variable. In the latter case, it can be minimized to reduce the deployment costs and become an objective of the problem. However, in our model, we do not explicitly define the minimization of N as an objective and rather use the concurrent optimization of coverage and interference to obtain a value of N that provides enough coverage of the building without inducing too much interference.

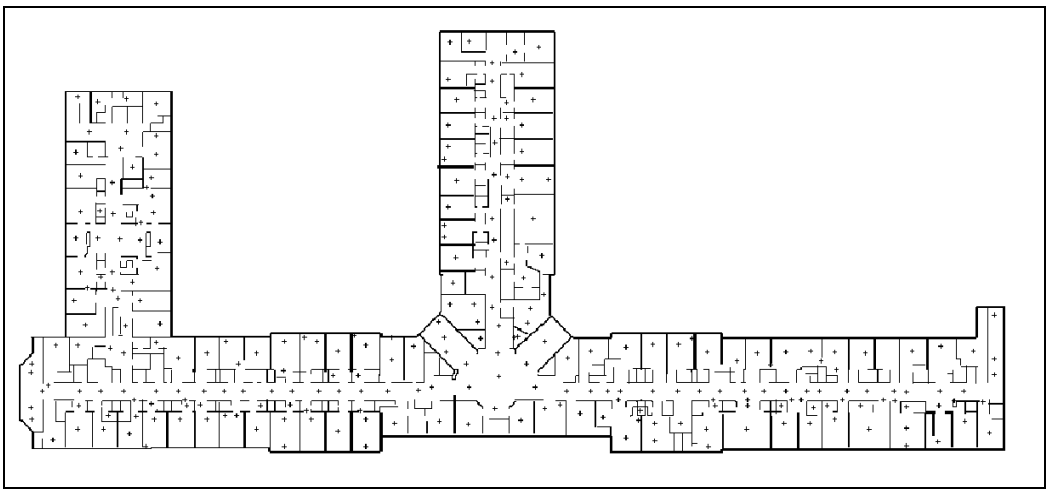


Fig. 1. Distribution of the $M = 256$ candidate AP locations on a 12400 m² building floor.

A set of M candidate AP locations is defined over the building floor as represented by the crosses in Figure 1. A solution S of the optimization problem is defined as a vector of M items $S = (s_1, \ldots, s_i, \ldots, s_M)$, each item s_i representing a candidate AP located at point i in the building. If the candidate AP at location i is selected in the solution S, s_i stores the transmission power p_i and direction of emission d_i. The item s_i is defined by:

$$s_i = \begin{cases} (p_i, d_i) & \text{if an AP is placed at the } i\text{th candidate location,} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Transmission power values $p_i \in \mathcal{P} = \{P_1, \ldots, P_{NP}\}$ and directions of emission to $d_i \in \mathcal{D} = \{D_1, \ldots, D_{ND}\}$ belong to two discrete sets of values. Given a directive antenna, each D_i represents a possible orientation of the main lobe of the radiation pattern. In the results presented in Section 5, an omni-directional antenna is used and thus d_i is fixed.

For each candidate location, a 2D coverage map is computed with a specific propagation simulator (De La Roche et al, 2007). A coverage map is defined as a set of mean received signal powers associated with rectangular areas defined by their top left-hand corner p=(x,y) and dimension s=(s_x, s_y) in pixels. For the sake of clarity, these areas are numbered from *1* to L, and referred to as the blocks B_l. The mean received signal power in dBm (logarithmic scale) from an AP numbered k on the block B_l is referred to as F_l^k.

Three optimization criteria are defined in (Jaffrès-Runser et al., 2008). The first one ensures that coverage is provided for all the blocks B_l of the design area. A block B_l is considered covered if F_l^k meets a given threshold. The second criterion ensures that the power of interference due to the APs that are not serving block B_l is limited. And the third quality of service (QoS) criterion ensures that the throughput available on block B_l is above a fixed threshold value.

For each criterion (coverage, interference mitigation or throughput), a specific utility value U_l is assigned to a block B_l. This utility value is derived from the power values F_l^k and depends on the kind of evaluation performed. For the coverage criterion, this value represents the maximum received power value on the block B_l. For the interference criterion, this utility value is the power of the second best AP received in the block (it is the strongest interfering signal). For the QoS criterion, the utility is the estimated throughput d_l obtained in that block B_l assuming a uniform distribution of N_r users in the building.

With such a definition, a good solution in terms of coverage is a solution where there is no lack of coverage. A good solution in terms of interference is a solution where the power of the signals that interfere with the main AP is minimized. A good solution in terms of QoS is a solution where the throughput provided to the user meets a minimum threshold value.

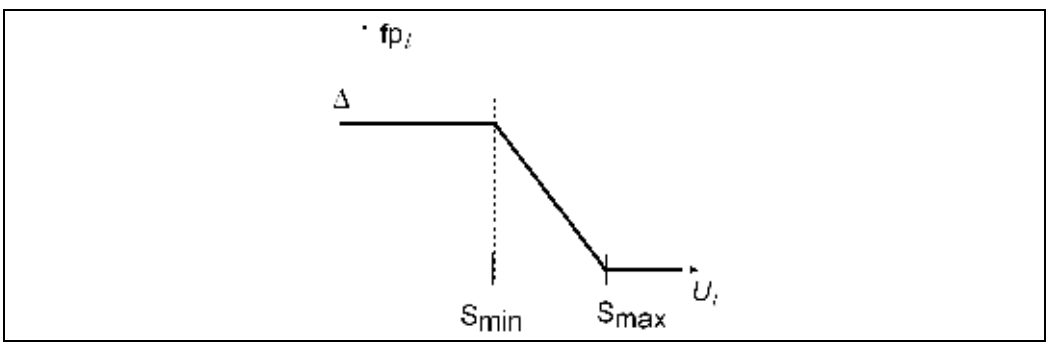


Fig. 2. Value of the penalty fp_l when the utility value U_l is minimized.

A penalty value fp_l estimating the quality of B_l regarding a specific criterion is computed. This value penalizes a block whose utility value U_l does not meet the constraints for that particular criterion (*cf.* (Jaffrès-Runser et al., 2008) for further descriptions). This penalty is a function of the utility value U_l on the block B_l and is depicted on Fig.2. When minimizing utility U_l, we have a penalty equal to 0 when U_l is higher than a threshold S_{max}. In this case, the constraint on U_l is fully met. A maximum penalty of Δ is applied when the utility becomes lower than the S_{min} threshold. A linear penalty is considered in between.

Each optimization criterion is defined as a quadratic weighted sum of the penalty values accounted for each block B_l:

$$f = \sqrt{\sum_{B_l, l \in \{1,\ldots,L\}} \mu_l . \mathrm{fp}_l^2}. \tag{2}$$

This quadratic representation allows us to concurrently minimize the average penalty together with the standard deviation values for the penalties, thus ensuring a more homogeneous distribution of coverage, interference and throughput. The notations f_{cov}, f_I and f_{QoS} are used in the following to address the coverage, interference mitigation and QoS provision criteria, respectively.

Each objective measures the quality of one feature of the network. Getting the solution that has the best possible rating for each criterion is barely possible, especially as the optimization criteria have an antagonistic influence on the solutions. For instance, networks made up of a high number of APs have good coverage and throughput performance but suffer from high interference levels.

3. Multiobjective optimization

3.1 Definitions

This section summarizes the main concepts of multiobjective optimization. First, we introduce the dominance relation between two solutions $\mathbf{x} \in S$ and $\mathbf{y} \in S$ of an n-objective MO problem in definition 1. Then, definition 2 defines the Pareto optimality and definition 3 the optimal Pareto front of a MO problem. The difference between the optimal Pareto front and the estimated Pareto front is presented in definition 4 and finally, definition 5 explains the notion of Pareto rank.

Definition 1: *A solution* **x** *dominates a solution* **y** *for a n-objective MO problem if* **x** *is at least as good as* **y** *for all the objectives and* **x** *is strictly better than* **y** *for at least one objective.* Mathematically, we have for a minimization problem:

$$\forall i \in [1, n] : f_i(\mathbf{x}) \leqslant f_i(\mathbf{y}), \quad \exists j \in [1, n] : f_j(\mathbf{x}) < f_j(\mathbf{y}) \tag{3}$$

Definition 2: *A solution* $\mathbf{x} \in S$ *is Pareto optimal if there is no other solution* $\mathbf{y} \in S$ *that dominates* **x**.

A Pareto-optimal solution is therefore a non-dominated solution of the problem.

Definition 3: *The optimal Pareto front* $\mathcal{F}^*$ *of a multiobjective problem is defined as the set of Pareto-optimal solutions (or the set of non-dominated solutions).*

The task of a multiobjective optimization search is to find the Pareto optimal front $\mathcal{F}^*$. The set of non-dominated solutions found at the end of the search represents the best approximation known for the Pareto optimal front. It is defined as follows:

Definition 4: *An estimated Pareto front* $\mathcal{F}_P$ *of a multiobjective problem is the set of non-dominated solutions obtained at the end of a search performed by any heuristic. The search succeeds if the estimated Pareto front is equal to the optimal Pareto front, i.e.* $\mathcal{F}_P \equiv \mathcal{F}^*$.

Figure 3 represents the optimal Pareto front obtained for a multiobjective minimization problem of two criteria. This figure is obtained after an exhaustive search of all the solutions of an instance of the WLP problem. As the search is exhaustive, the estimated Pareto front is equal to the optimal Pareto front.

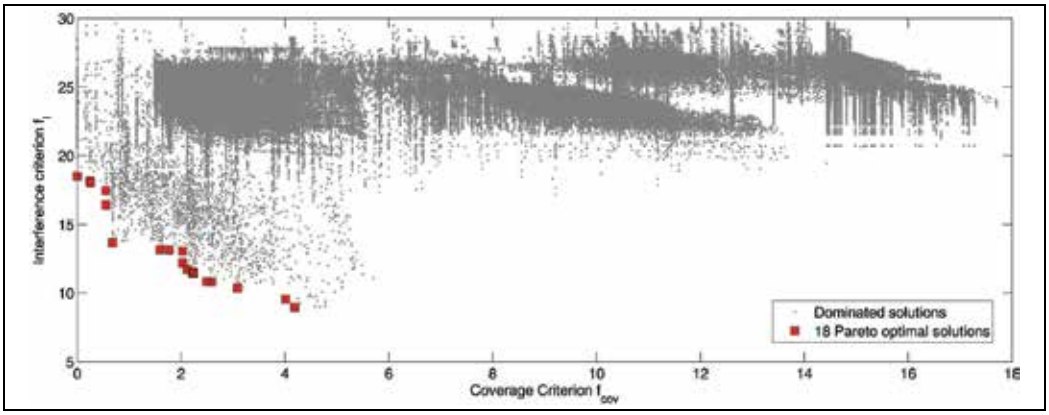


Fig. 3. Illustration of the Pareto set for a 2-objective minimization problem: the WLP problem with coverage and interference mitigation criteria.

In this problem instance, the coverage and interference mitigation criteria are minimized for a small deployment environment of 2100m². There are M=129 candidate AP sites and the number of APs to plan is fixed to N=3. Each point on this graph represents the evaluation of one solution for both criteria. The black points are the solutions dominated by the solution of the Pareto front. The desirable solutions are all the solutions depicted with red squares which provide the best available trade-offs between coverage and interference mitigation. The solutions of an MO problem can be sorted according to their Pareto rank defined as follows:

Definition 5: *The rank of a solution* $\mathbf{x}$ *is defined by* $R(\mathbf{x}) = 1 + d(\mathbf{x})$, *where* $d(\mathbf{x})$ *is the number of solutions by which* $\mathbf{x}$ *is dominated in the set of feasible solutions S. The solutions of the theoretical Pareto front have a rank* $R(\mathbf{x}) = 1$.

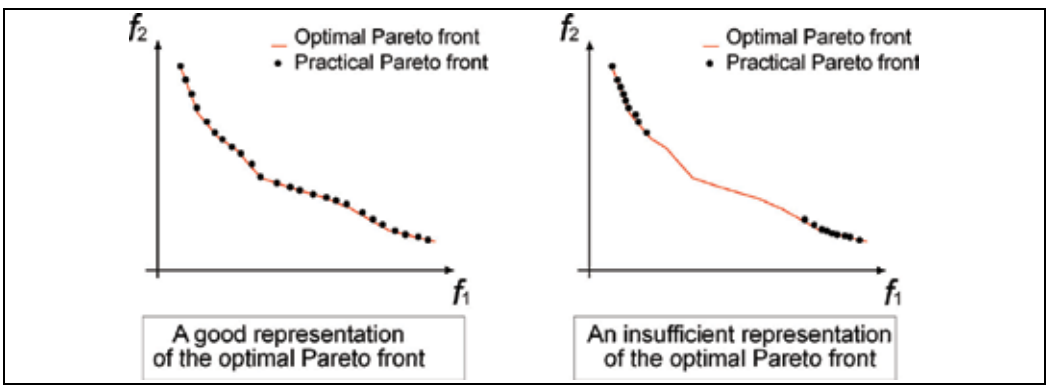


Fig. 4. Good and insufficient representation of the optimal Pareto front.

It is not always possible and necessary to search for all the solutions of the Pareto front. However, the designer needs a good representation of the Pareto front to take its decision. Therefore, heuristics have to provide an estimated Pareto front that displays the main trade-offs of the optimal front. To that end, the estimated front must be made of a set of solutions that evenly spans the whole front as depicted in Figure 4. If the search only concentrates on some specific parts of the search space, only isolated parts of the Pareto front can be reached, which may not be good candidates for a designer's choice.

3.2 Multiobjective metaheuristics

This section introduces the main stream work available in the field of multiobjective optimization. We also refer the reader to a survey description of multiobjective optimization in (Collette & Siarry, 2004) and a more detailed one on evolutionary MO optimization in (Deb, 2001) for further investigations.

MO optimization techniques can be classified into the 3 following types of algorithms (Van Veldhuizen, 1999):

1/ *A priori* search techniques: A standard mono-objective search algorithms is applied to a single evaluation function that is a weighted sum of all the optimization criteria. The designer chooses the weight to assign to each evaluation function. The heuristic only provides one solution at the end of the search, reflecting the trade-off induced by the *a priori* choices of the designer when setting the weights of the optimization criteria.

2/ *A posteriori* search techniques: A specific MO heuristic looks for the best possible approximation of the optimal Pareto front during the search. The next step is the choice of the final solution by the designer. When the estimated Pareto front is composed of too many solutions, another selection / sorting step is needed to present only some relevant solutions to the designer for a final choice.

3/ Progressive techniques: In these techniques, the designer directly interacts with the algorithms during the search. The search is composed of a sequence of decision making cycles (where the designer input its preferences/constraints) and search cycles. A search cycle may either use an *a priori* or an *a posteriori* MO search algorithm.

The Tabu based MO optimization algorithms listed in the review paper of Jones et al. (Jones et al., 2002) are *a priori* techniques. The drawbacks of such techniques are twofold. Firstly, obtaining the trade-off targeted by the designer by choosing the weights assigned to the criteria is not always trivial. The dynamics of the optimization criteria (gradient, order of magnitude) have to be known in advance to adjust the weights properly. Depending on the complexity of the criteria, such information is not always available in real-world implementations. As a consequence, empirical trials are needed to determine the appropriate weights leading to the solution producing the desired trade-off. Secondly, for concave Pareto fronts, there may be regions of the front that are not defined by a combination of weights, and consequently certain combinations of weights represent two points on the front (Fonseca & Fleming, 1995).

Jones et al. (Jones et al., 2002) clearly demonstrated the prevalence of genetic based approaches. One reason for this is that some important theoretical algorithms have been developed in the 1980s and early 1990s for the approximation and generation of the optimal Pareto front by genetic-based methods. These include N.S.G.A., the Non-dominated Sorting Genetic Algorithm (Srinivas & Deb, 1994) and M.O.G.A., the Multi-Objective Genetic Algorithm (Fonseca & Fleming, 1998). The enhanced version of the Non-dominated Sorting Genetic Algorithm, called N.S.G.A.-II (Deb, 2002), is now recognized as one of the leading algorithms in the domain.

For these genetic based algorithms, the solutions of the population are evaluated based on a dominance metric and the non-dominated ones are stored in the estimated Pareto front. Crossover and mutation strategies are implemented in the same way as for mono-objective search to favor exploration and intensification.

The MO version of a genetic metaheuristic differs from its standard mono-objective version by the selection of the new population. This selection relies on a unique function to sort out the good solutions from the poor ones. For MO problems, the efficiency of each individual is

represented by a vector of criteria. Ranking all the individuals of the population is done by a fitness function which reflects the values of all the criteria of a solution. For instance, in M.O.G.A, the solutions are ranked using an affine transformation of the Pareto-rank metric (**Definition 4)** between 0 and 1. The drawback of such a fitness function is that is does not yield a good diversity of the Pareto front.
Diversity is enforced in the fitness function of the N.S.G.A. algorithm (Srinivas & Deb, 1994) by adding a niching operator to the fitness function. This niching operator favors the selection of a solution whose evaluation belongs to an unexplored part of the current set of non-dominated solutions. This operator involves a sharing factor σ_{share} that sets the extent of the sharing in the problem, i.e. how far any two solutions are considered to share the same fitness. The main improvement provided by N.S.G.A.-II in (Deb, 2002) is to avoid the explicit setting of σ_{share} by providing a comparison operator that accounts for both the Pareto rank and the average distance in the function space between a solution and its neighbour non-dominated solutions.

4. Multiobjective Tabu (MO-Tabu)

4.1 Tabu heuristics for multiobjective Ooptimization

As presented in the previous section, there is a widespread interest within the engineering design community in applying multiobjective genetic algorithms to real-world problems. However, genetic algorithms can experience difficulties on highly constrained problems. Tabu search, thanks to the local search heuristic at its heart, can navigate highly constrained search spaces successfully. A multiobjective variant of Tabu search is therefore a valuable tool for engineering design. In this review, we are focusing on the *a posteriori* MO Tabu search methods where an estimation of the optimal Pareto front is targeted.
Interest for Tabu search in Multiobjective optimization has increased in the last decade (Hansen, 2000; Gandibleux & Freville, 2000; Ho et al., 2002; Armentano & Arroyo, 2004; Choobineh et al., 2006; Baykasoglu et al, 2006; Kulturel-Konak et al., 2006; Jaffrès-Runser et al., 2008; Jaeggi et al., 2008). All these techniques adapt the mono-objective Tabu search heuristic by proposing several modifications:

- Firstly, the search algorithm does not store a single estimate of the optimal solution, but a set of all the non-dominated solutions encountered during the search. At the end of the search, this set represents the estimated Pareto front. This feature is common to all the presented solutions in this section.
- Secondly, as there is not a single evaluation function to decide on the quality of the neighbour solutions, a new strategy for choosing the best neighbour at the end of one iteration has to be set.
- Thirdly, the Tabu lists are adapted to a multiobjective formulation, too. For instance, a Tabu list can store Tabu functions to avoid searching solutions along criteria already well explored (Gandibleux & Freville, 2000).
- Lastly, diversification techniques are adopted to increase the quality of the representation of the final Pareto front.

All the works presented here can be separated into two categories:
The first ones explore the space by using a single Tabu search path (Gandibleux & Freville, 2000; Ho et al., 2002; Kulturel-Konak et al., 2006; Baykasoglu et al, 2006; Jaeggi et al., 2008) while the other ones launch several Tabu searches in parallel (Hansen, 2000; Armentano & Arroyo, 2004; Choobineh et al., 2006; Jaffrès-Runser et al., 2008).

In the first case, the structure of a Tabu Search iteration is similar to an iteration of the standard mono-objective algorithm as presented in Figure 5, except for the selection of the new estimated Pareto front $\mathcal{F}_P^{new}$. This selection relies on a Pareto dominance test to get the non-dominated solutions from the set composed of the current Pareto front $\mathcal{F}_P$ and the set of neighbour solutions $\mathcal{V}(S_c)$.

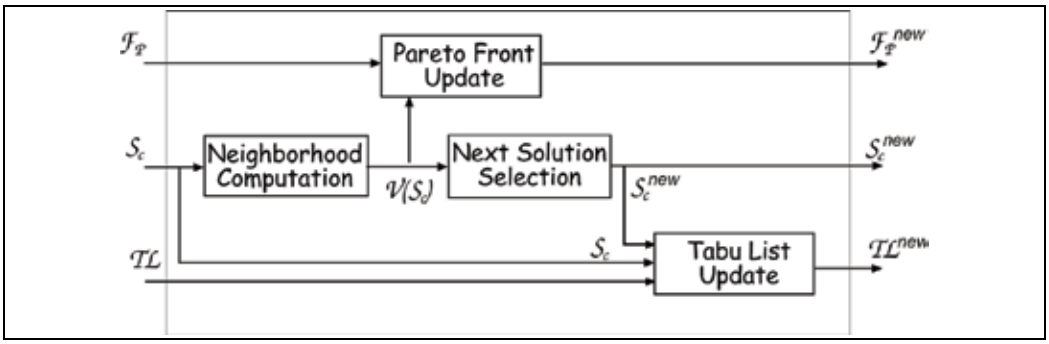


Fig. 5. Structure of an iteration for a multiobjective Tabu Search when only one search path is exploited

In the second case, there is a pool of solutions representing a search front $\mathcal{F}_c$ which is analogous to the search front defined in an evolutionary MO metaheuristic. As depicted in Figure 6, the search front is expanded through a neighbourhood search, whose solutions are evaluated and added to the estimated Pareto front to compute $\mathcal{F}_P^{new}$. Then a new search front $\mathcal{F}_c^{new}$ is selected. For these parallel strategies, the goal is to orient the parallel searches towards different parts of the Pareto front. Therefore, the selection of the new search front should orient the search to under-explored parts of the Pareto front. If p search paths are considered, p Tabu lists have to be updated to avoid a cyclic search for each path.

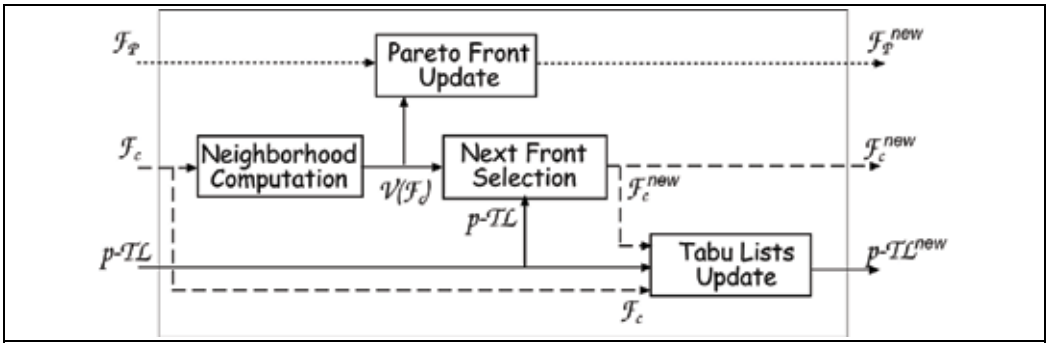


Fig. 6. Structure of an iteration for multiobjective Tabu Search when m parallel Tabu search paths are exploited

Choice of the new search solution

The first step of a TS iteration is the creation of a set of neighbour solutions of the current search solution S_c. In a mono-objective problem, the new current solution S_{new} is chosen as the neighbour solution that optimizes the unique optimization criterion. As a matter of fact, such a choice is not directly applicable to multiobjective problems. The whole point in a MO

Tabu search implementation is to determine what the next search step is and implicitly, which strategy will bring the search closer to unexplored parts of the Pareto front. Two kinds of strategies for the selection of S_{new} have been proposed so far.

In the first strategy, a search direction in the function space is set and the solution of the neighbourhood that optimizes the objectives along that particular direction is selected. The search direction is defined by a set of weights $\{ \lambda_k, _{k \in [1..m]}\}$ used to optimize a weighted sum of the m criteria $f_{k, k \in [1..m]}$ defined by:

$$f(S) = \sum_{k=1}^{m} \lambda_k f_k(S) \tag{4}$$

For instance, Kulturel-Konak et al. (Kulturel-Konak et al, 2006) propose to randomly choose a function f_k and to select the best neighbour for this particular criterion. In this case, the search direction is defined by the function for which all the weight values are equal to zero, except for the selected function which gets a weight of 1. With such a strategy, the set of search directions is equal to m, which necessitates more moves and therefore more function evaluations if the optimal search direction is a linear combination of the objectives. In the same way, Choobineh et al. (Choobineh et al., 2006) select a criterion as the search direction but instead of 'time-multiplexing' them, they perform a parallel Tabu search where each path optimizes one of the m objectives.

Several works have focused on adjusting the weights at each solution selection to favor a rapid convergence to the Pareto front and drive the search towards unexplored parts of the front. Gandibleux and Freville (Gandibleux & Freville, 2000) favor the optimization of functions that have seen little improvement in the previous iteration. In the same fashion, Hansen (Hansen, 2000) puts more emphasis on the functions that have seen few improvements in the current search front. The difference with the implementation of Gandibleux and Freville is that Hansen also scales the weights proportionally to the number of solutions of the current search front that suffer from such a small improvement.

Armentano and Arroyo (Armentano & Arroyo, 2004) arrange the solutions of the Pareto front into clusters. Each one of their parallel Tabu search is oriented to the centroid of one of their clusters to get a good representation of the Pareto front.

The second strategy selects a solution by relying on a Pareto dominance criterion. Baykasoglu (Baykasoglu, 2006) constructs the new Pareto front $\mathcal{F}_P^{new}$ and then randomly selects the next solution in it.

Ho et al. (Ho et al, 2002) sort all the solutions of the search front according to their Pareto rank and apply a fitness function that favors the selection of non-dominated solutions whose neighbourhood is less dense in the function space.

The definition of the Tabu lists

All the heuristics use a regular Tabu list that stores for a given amount of iterations the previous solutions or movements to avoid cycling in the search. However, additional Tabu lists can also be introduced to benefit the search in the MO context.

As proposed by Gandribleux and Freville (Gandibleux & Freville, 2000), a second Tabu list is created to prevent the search along criteria that experienced consequent improvements in the past. The rationale for defining such a Tabu list is to prevent the algorithm from searching along the steepest descent gradient.

A Tabu list called the 'Intensification Memory' (IM) has been introduced in the implementation of MO Tabu by Baykasoglu et al. in 1999 (Baykasoglu et al., 1999). This memory stores non-dominated solutions that have not yet been selected as a current search solution yet. When several non-dominated solutions exist in the neighbourhood of S_c, the IM list is updated with the non-dominated solutions that are not chosen to become S_{new}. When a neighbourhood does not contain any non-dominated solution, the oldest element of the IM list is used as a new solution. Upon update, a solution is stored in the IM only if it is not dominated by any solution of the neighbourhood, by any solution of the IM list or by any solution of the estimated Pareto front. A solution of the IM list is added to the Pareto front once it has been selected as a new current solution S_{new}. This intensification memory list provides a tool to enforce local exploration of the solution space before storing a non-dominated solution in the estimated Pareto front.

The same memory structure has been implemented by Baykasoglu in (Baykasoglu, 2006) and by Jaeggi et al. in (Jaeggi et al.). Instead of choosing the oldest solution in the IM list, Jaeggi et al. select a random solution when intensification is needed. When the IM list is empty, Jaeggi et al. add a diversification step that randomly picks a point in the estimated Pareto front for S_{new}.

When p parallel Tabu searches are performed, p Tabu lists are maintained to avoid cycling during the search for each individual path (Hansen, 2000; Armentano & Arroyo, 2004; Choobineh et al., 2006; Jaffrès-Runser et al., 2008).

Diversification techniques

The search for the optimal Pareto front is challenging and diversification techniques are recommended when a single Tabu search path is considered. In this case, a rather small subpart of the solution space is explored within an iteration, resulting in an increased probability of limiting the search to a local subspace of the solution space. Hence, it is important to restart the search timely when no improvement of the Pareto front is noticed.

For instance, Kulturel-Konak et al. (Kulturel-Konak et al., 2006) restart the search with a random solution if the Pareto front remains unchanged for the last ($I_{max}/4$) iterations, with I_{max} the maximal number of iterations. The Tabu list is cleared and the search resumed with the new random solution. We would like to point out that restart strategies do not necessarily induce diversification. For instance, the restart procedure proposed by Baykasoglu (Baykasoglu et al., 1999; Baykasoglu, 2006) does not introduce diversity since the new starting solution is chosen among non-dominated solutions. Therefore, only downhill moves are allowed, resulting in a local convergence.

Search diversification may also be implicitly favored through the selection of the new current solution S_{new}. This is the case when the search direction is adapted by changing the weights assigned to the criteria ((Gandibleux & Freville, 2000 ; Hansen, 2000). Since the weights are adapted along the search to avoid the exploration of neighbourhoods with already good performance, premature local convergence is less likely to occur.

When parallel Tabu search paths are considered (Hansen, 2000; Armentano & Arroyo, 2004; Choobineh et al., 2006; Jaffrès-Runser et al., 2008), no specific diversification procedures are required. Indeed, the number of parallel searches intrinsically increases the amount and the diversity of the solutions tested at each iteration.

Tabu search is a promising metaheuristic that efficiently tackles Multiobjective optimization. In the recent work of Jaeggi et al (Jaeggi et al., 2008), the proposed MO Tabu search implementation (called PR-MOTS) is compared to the leading genetic MO heuristic NSGA-II for a set of five standard test functions. PR-MOTS performed similarly to NSGA-II,

providing a greater likelihood of improvement in the objective functions but exhibiting a higher performance variability. Moreover, the local search component and the flexibility of handling constraints for a large number of variables makes Tabu Search a particular attractive metaheuristic for multiobjective optimization.

4.2 MO-Tabu

A simple MO Tabu Search implementation presented in (Jaffrès-Runser et al, 2007) is described in the following to highlight the assets of Tabu algorithms for MO search. It is referred to as PMOTS, standing for 'Parallel-MultiObjective Tabu Search'.

The algorithm detailed in Figure 7 exploits K parallel search paths and stores the non-dominated solutions in an estimated Pareto front $\mathcal{F}_P$. A Tabu list TL_k is assigned to each search path $k \in [1,..,K]$. The duration of a Tabu list is updated at each iteration with a random size $t \in [T_{min}, \ldots, T_{max}]$.

```
1: Select the first search front F_c(0) made of K solutions;
2: Init the K Tabu lists TL_k=∅, k ∈ [1,..,K];
3: Init the estimated Pareto front F_P=∅;
4: For each iteration i in [0,..,I_max] do:
5:    Init the next search front F_c(i+1)=∅;
6:    For each solution S_k of the current search front F_c(i) do:
7:          a) Compute and evaluate the neighbourhood set V(S_k);
8:          b) Select from V(S_k) the solutions with Pareto rank
           R(S) ≤ R_max and store it as the set P_R(S_k);
9:          c) Select randomly a solution of P_R(S_k) and add it
           into the new search front F_c(i+1);
10:         d) Concatenate P_R(S_k) with the Pareto front F_P;
11:         e) Update the Tabu list TL_k;
12:   End;
13:   Remove the solutions having rank R(S)>1 from F_P;
14: End;
15: Return F_P;
```

Fig. 7. Macro-algorithm of PMOTS

In the above algorithm, the K parallel search paths are represented as a search front $\mathcal{F}_c(i)$ of K solutions. A new search front $\mathcal{F}_c(i+1)$ is selected in each iteration by choosing promising solutions that are not always non-dominated to avoid a premature convergence of the algorithm. Therefore for a path k, each new solution is selected randomly in the set of neighbour solutions $\mathcal{V}(S_k)$ of S_k having a Pareto rank $R \leq R_{max}$. In this algorithm, the Pareto ranking is local to the set of neighbour solutions and does not include the current estimated Pareto set. By not including $\mathcal{F}_P$ and selecting fairly good solutions with the Pareto rank constraint, diversity is introduced within the search strategy.

The rationale behind implementing a parallel Tabu Search is the following. By properly choosing the set of initial solutions and the neighbourhood construction strategy, the optimization problem can be reduced into sub-problems being solved concurrently. Ideally, if the variable search space can be split into K subsets and if the neighbourhood construction strategy of each subset is able to generate all the solutions of this particular subset, then the combination of all the K parallel search paths provides a regular representation of the Pareto front.

However, each subset does not contribute equally to the Pareto front representation. A first subset might contain half of the non-dominated solutions, while another one might only have one or two Pareto optimal solutions. Therefore, if the search in the second subset is restricted to the only variables of the subset, the ratio between the cost (in terms of computation effort) and the gain (in terms of the number of solutions of the Pareto front obtained) is very poor. Therefore, to provide a more efficient search, the neighbourhood definition should not be limited to a subset of the search space, but provide some chances to move from one subset to another and thereby intensify the search in more promising subsets of the search space. The whole point in such a simple parallel optimization algorithm resides in the definition of the subsets and the neighbourhood construction process.
The first search front $\mathcal{F}_c(0)$ is composed of K solutions randomly chosen within each subset. Depending on the MO problem considered, the neighbourhood construction process must provide most of the solutions within its own search subset and only few solutions that belong to other subsets. If a neighbour solution that belongs to another subset is promising, the search may move to that subset. This is the case when this promising solution has a Pareto rank smaller than $R_{\max}$ and is selected as the new search solution.
The next section presents two implementations of PMOTS. In the first one, we present the WLP problem, and practical considerations relative to the exploitation of the estimated Pareto front are addressed here. Then, we broaden our presentation of MO-Tabu applications by discussing the problem of benchmarking the routing performance in wireless sensor networks.

5. Application to wireless networks optimization and evaluation

5.1 PMOTS for the WLP problem

PMOTS has been proposed to solve the WLP problem in (Jaffrès-Runser et al, 2007). We recall that in this problem, a solution is defined by a vector of M items $S = (s_1, \ldots, s_i, \ldots, s_M)$, each item s_i representing a candidate AP location. If the candidate AP at location i is selected in the solution S, s_i stores the transmission power p_i and direction of emission d_i of the AP. If the AP at location i is not selected in the solution, s_i =0. There is a discrete set of N_P possible transmission powers and a discrete set of N_D possible directions of emission. At the beginning of the search, d_i and p_i are set to an initial value for all candidate APs.
Neighborhood construction strategy:
In this implementation, a neighbour of a solution S is constructed by following either one of these moves:

- ***Swap move***: a selected AP at position i is deselected and a deselected AP at position j is selected. d_i and p_i values remain the same as the ones already stored in the item s_i.
- ***Addition move*****:** A new AP is selected in the solution S,
- ***Delete move:*** A selected AP is deselected from the solution S,
- ***Power change move:*** Change p for a selected AP to an admissible power different from the current value of p,
- ***Direction change move***: Change d for a selected AP to an admissible direction of emission different from the current value of d.

When computing all the neighbours according to these rules, we get the following number of neighbours as a function of the number N of currently selected APs of solution S:

- N(M-N) solutions with a swap move,
- M-N solutions with an addition move,
- N solutions with a delete move,
- N(N_P -1) solutions with transmission power change move,
- And N(N_D -1) solutions with direction change move.

The solution space can be divided into subsets by gathering the solutions with the same number N of selected APs into a same subset. Consequently, a subset is defined by the number N of selected APs of its solutions and the solution space is divided into N subsets. A swap move, a transmission power change and a direction change move keep the same number of selected APs in the resulting neighbour set. The addition and deletion moves increase or reduce N, and therefore provide neighbours that belong to other subsets. The proportion of all the neighbours that belong to the same subset compared to the number of neighbours that do not belong to the same subset modifies the chances for a search path to continue to explore the old subset or to change subset.

There is an intensification of the search when the search path stays in the same subset and an exploration of the search space when the search changes subset.

For the building plan represented in Figure 1, we have M=256 candidate AP locations. There are also N_P=5 possible transmission power values and N_D=4 possible directions of emission. For this real-world implementation, how does the neighbourhood definition favor the exploration or intensification of the search depending on the number of selected neighbours N?

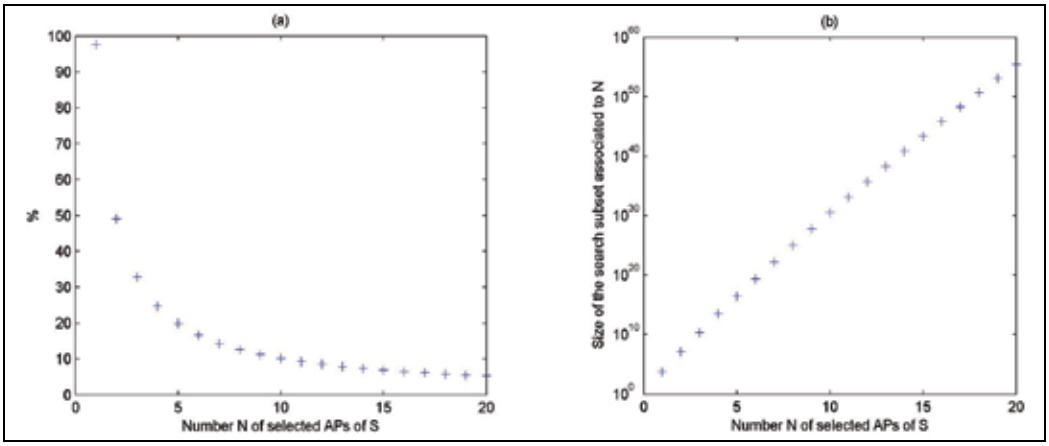


Fig. 8. Analysis of the neighbourhood $\mathcal{V}(S)$ of a solution S: (a) Percentage of neighbour solutions $\mathcal{V}(S)$ that belong to another subset as a function of N, the number of selected APs from the origin solutions S. (b) Size of the subsets having N selected APs as a function of N.

Figure 8-(a) displays the percentage of neighbours that belong to another subset depending on the number N of selected APs of the origin solution S. For small values of N, the percentage of neighbours with a higher number of APs is high. In this case, it is very likely that a search path changes to a subset with a higher number of selected APs. For larger values of N, there are about 5% of the neighbours that belong to other subsets, reducing the chances for a path to change subset during the search and increasing the exploration of the subset.

To summarize, for small values of N, exploration is favored while for larger values of N, intensification is favored.

Figure 8-(b) represents the size of the subsets obtained for N on a logarithmic scale varying from 1 to 20. The size of each subset increases drastically with N. Since the subsets are smaller when N is low, it makes sense to promote exploration, as a single neighbourhood construction already tests a good part of the subset. Fewer searches are here necessary in this case to explore the subset. When N increases, intensification is enforced as there are more solutions to test in a subset.

Tabu list

The Tabu list is a list of candidate APs. When a candidate AP is added to the Tabu list, it stores the current values of p_i and d_i. For a swap move, the list stores the AP that is no longer selected in the solution. When the new solution results from an AP addition, the Tabu list stores a specific 'fake' AP that signifies that the addition move is now taboo. In the same way, when the solution results from the deletion of an AP, the Tabu list stores another 'fake' AP that signifies that the delete move has become taboo. When these add and delete moves are taboo, the search is forced to intensify in the current subset, which is beneficial in order to avoid too much exploration. Upon a transmission power or direction change, the candidate AP with the old value of p_i or d_i is stored in the list.

About the initial solution and the number K of parallel search paths

As there are M different subsets defined for the WLP planning problem, a first choice would be to use K=M search paths in the PMOTS algorithm. As a matter of fact, subsets composed of solutions with a high number of selected APs N can not provide very good solutions. Based on practical considerations, a planning solution that involves more than 30 APs would not perform optimally for the building described in Figure 1. For such a high number of deployed APs, the intensity of the interference in the building results in a very poor interference criterion and a reduced QoS performance. Complete coverage of the building can be achieved for N=4 APs but for a very low throughput.

Thus, we set a value of K=15 different parallel search paths with a first search path launched in the subset made of N=3 selected APs, the k^{th} search path in the subset N=k+3 and the last search path in the subset made of N=18 selected APs.

The first search front is composed of the solutions that are the starting points for the K search paths. Each initial solution is composed of a different number of selected APs to start the search in a given subset. Consequently, the initial solution of path k presents k+3 APs. The location of each selected AP in an initial solution is selected randomly in the set of candidate AP locations.

Selection of the solutions of the Pareto front

The estimated Pareto front obtained after 300 iterations is composed of 148 solutions that are presented in Figure 9. The coverage, interference mitigation and QoS criteria are optimized in this search. The target throughput for a uniform distribution of 200 users is set to 256kbits/s. For smaller instances of the WLP problem, it is shown in (Jaffrès-Runser et al, 2008) that a good approximation of the optimal Pareto front is obtained with PMOTS. In Figure 9 we have the best solutions found so far, trading-off all the planning criteria. There are 89 solutions out of 148 that have a perfect coverage and provide different trade-offs between interference and QoS. For this problem instance, PMOTS has evaluated an average of about 40000 solutions per iteration.

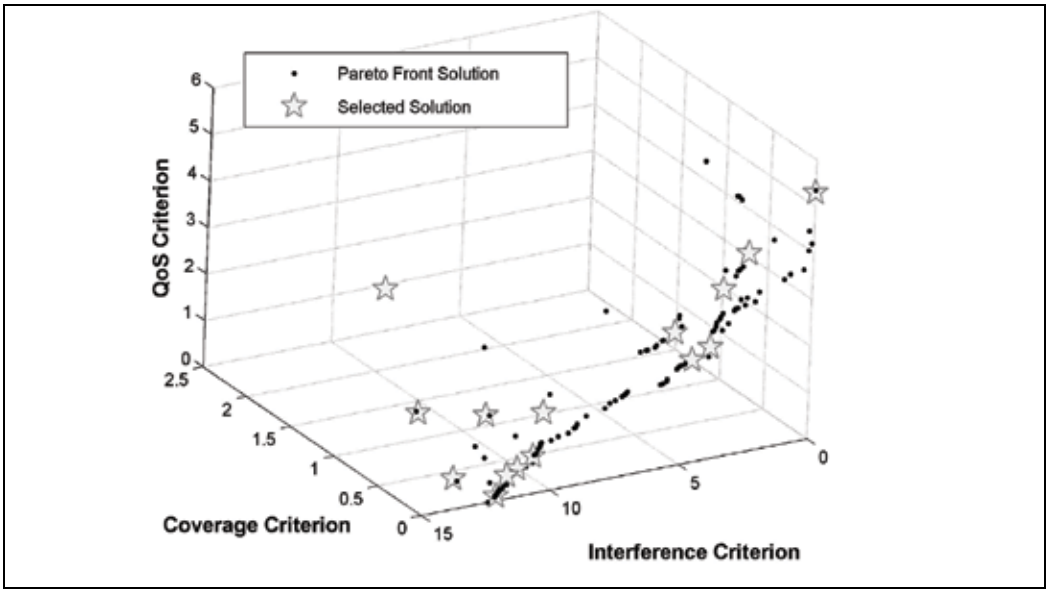


Fig.9. Representation of the estimated Pareto front for the WLP problem with M=256 candidate locations after 300 search iterations. The 15 solutions selected after the PMOTS search are represented with stars.

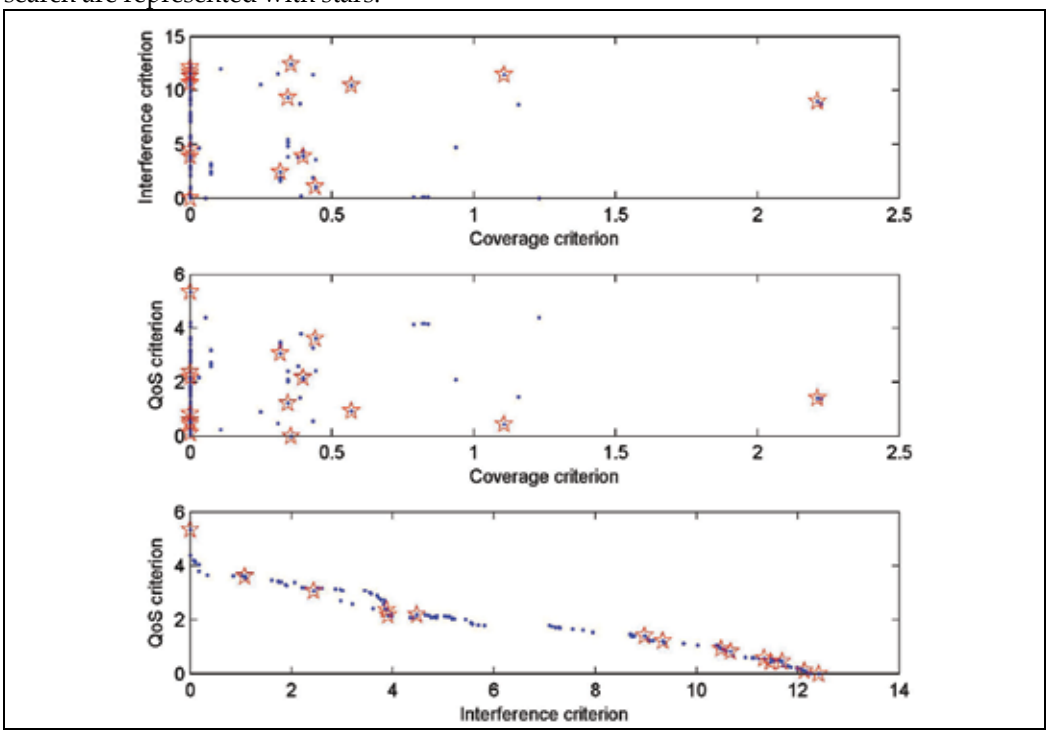


Fig.10. Projections of the estimated Pareto front for the WLP problem with M=256 candidate locations after 300 search iterations. The red stars represent the 15 solutions selected from the Pareto front.

Since the designer can not evaluate individually each one of the 148 solutions, a pre-selection of the solutions of the Pareto front is needed. An algorithm is proposed in (Jaffrès-Runser et al., 2008) which concentrates on the selection of N_F solutions representing significant and different trade-offs between the objectives. The selection is based on both the dissimilarity of the criteria trade-offs obtained and the dissimilarity of the selected solutions. The N_F=15 selected solutions are depicted on Figure 9. The projection of the Pareto front is given in Figure 10. The 15 selected solutions are represented by little red stars.

Three network planning solutions out of the 15 selected ones are presented in Figure 11. Upon analyzing the 15 selected solutions, we concluded that the networks obtained when the number of APs is low (i.e. N<13) are very promising as they show evenly distributed APs. The solutions selected with a higher number of APs (i.e. N≥13) present an uneven distribution of APs (*cf*. Fig. 11-(c)).

Among the solutions using 19 APs, the solutions with a nonuniform distribution of nodes present a lower interference criterion than the solutions with 19 evenly distributed APs. This is due to the fact that, in the first case, strong interference is localized in a smaller surface area, thus reducing the impact on the whole building. However, as shown by the low QoS criterion, the available throughput is high even though strong interference is generated by the APs. This is due to the fact that the transmission channels are not assigned during the planning stage and therefore, the throughput estimations can not completely account for the interference distribution in the building.

Adding the channel optimization variables would increase the problem's complexity. In our case, the channel assignment is performed after the planning stage, in a separate network optimization stage. The role of the interference criterion in the planning stage is to select solutions that simplify the channel assignment step. The solution with 19 APs sees its QoS criterion rise from 0.1 to 1.0 after channel assignment as it completely accounts for interference in the throughput computation. With the increase in computation power, it is clear that including the channel assignment into the planning problem formulation will benefit the overall quality of the solutions.

However, the search space subset for N=19 APs is also bigger than the subset for lower values of N. Therefore, it might also be that the search did not found the best trade-offs that belong to this subset and that a longer search would have been beneficial for that particular path.

Introducing some interaction between the search paths should improve the performance of PMOTS. From time to time, an intensification procedure could be started where the new solution S_{new} for a path k with bad performance can be chosen in the neighbourhood set of promising path j. Every time such an intensification procedure will occur, the performance of the K paths will be evaluated in terms of the number of solutions each one of them added to the estimated Pareto front since the last intensification procedure has been performed. Upon change, the Tabu list of path k would be erased as the old moves do not make sense anymore in the new search subset. By adding such an intensification procedure, we would really take advantage of the parallel processing of the PMOTS algorithm.

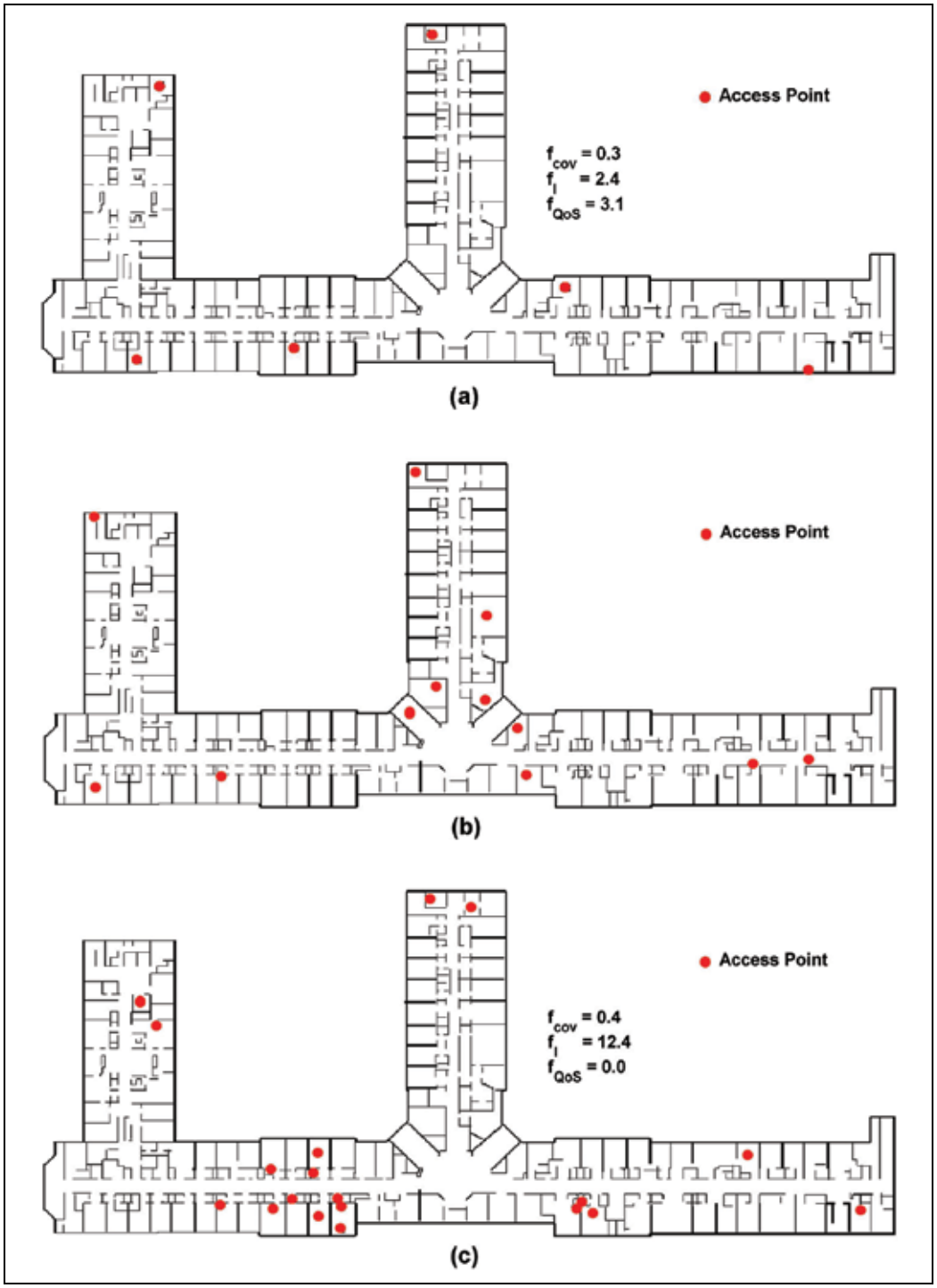


Fig. 11. Three solutions of the Pareto front: (a) Solution with 6 APs, (b) Solution with 12 APs and (c) Solution with 19 APs.

5.2 Application to the Evaluation of wireless sensor networks

PMOTS is adapted herein to address the problem of the evaluation of wireless sensor networks. The objective here is to quantify the achievable transmission performance of a Wireless Sensor Network (WSN). Our aim is to retrieve the Pareto optimal data forwarding patterns in the network with respect to three performance metrics: transmission robustness, transmission delay and energy consumption.

WSNs are composed of sensors equipped with a wireless transmission interface. These sensors report their sensed data to a specialized node or access point called the sink node through their wireless interface. The size of the network varies depending on the network application. Vast fields of sensor networks can be used to periodically report the activity in the field. Smaller networks can be deployed in buildings where they may sense temperature, track some target device or send alarms upon intrusion detection. Sensors are usually powered by batteries, and thus the energy consumption represents an important design criterion. Since the wireless interface consumes a significant amount of energy for listening, transmitting and receiving the data, the wireless routing protocol has a strong impact on the life duration of the network.

In this example, we introduce a model for analyzing the performance of routing in a wireless sensor network within a multiobjective framework. This model provides a tool to characterize the tradeoffs between robustness, delay and energy performance objectives depending on the network topologies and the transmitted traffic. When wireless sensors are deployed in outdoor areas, they become prone to node and transmission failures. Therefore, reliability of data transmission is a key performance metric of the network. As the data travels over the network, the average transmission delay should not affect the timeliness of the data arriving at the sink. If the transmission in the network takes too long, the data might not be accurate anymore, which results in a waste of energy.

The multiobjective optimization model

In this problem, a WSN consisting of N uniformly distributed sensors over an infinite plan is considered. It is assumed that the sensors are independent and randomly distributed according to a random point process of density ρ over the space $\Re^2$. A set of N_A sensors that belong to a circular area of radius R is defined as the communicating nodes of the network. These nodes can transmit, receive or forward data. The other nodes can only participate in the forwarding effort but can not be the final destination or send new data. A communication pattern is defined which is given by a set $\mathcal{S}$ made of S source nodes and a set $\mathcal{D}$ made of D destination nodes. In a WSN, we usually have D<<S as only few sink nodes exist.

The purpose of this model is to determine, given a network of density ρ and a communication pattern, what kind of trade-offs arise between the transmission robustness, the transmission delay and the energy consumption depending on the routing strategy involved. Routing between source and sink can use a direct transmission (single-hop) or a multi-hop path where intermediary sensors forward the data towards the sink. In some cases, a multi-hop relaying strategy may benefit the overall energy consumption of the network as less power is needed to transmit the information on short range hops than on a single long-range hop. However, the more hops are used on a path, the longer the transmission lasts. There is a clear trade-off between the energy consumption and delay. As

stated previously, transmission robustness is also of a great concern in such networks. Therefore, transmission redundancy may be also introduced by defining multiple source destination paths. Routing in a WSN is modelled as a multiobjective problem in the following. The optimization criteria are the robustness, the delay and the energy consumption of a continuous flow of data between a set of sources and a set of sink nodes.

Optimization variables

Each sensor node of the network can retransmit a received data packet with a given probability. This forwarding decision is modelled with a discrete variable $\mathbf{x}_i$ representing the probability that a node forwards a message by broadcasting it. The set of P possible values for $\mathbf{x}_i$ is given by $\{p_k$, for $k \in [1, \ldots, P]\}$ with $p_1 = 0$ and $p_P=1$. If P=2, $\mathbf{x}_i$ is a binary decision variable and a sensor simply decides to broadcast a packet or not. A solution is given by the set of broadcasting probabilities:

$$\mathcal{S} = \{x_i\}_{i \in [1,..,N]} \tag{5}$$

The aim of the optimization model defined in this work is to see how the selection of the relaying nodes impacts the robustness/energy/delay tradeoffs in the WSN.

We consider that all the nodes with $\mathbf{x}_i > 0$ constantly transmit data. For a link between two nodes i and j, the expected interference I_{ij} created at a node j can be computed by weighting the sum of the powers received at node j by all the other transmitters of index $k \neq i$ by the probabilities of forwarding $\mathbf{x}_k$:

$$I_{ij} = \sum_{k=1}^{N} P_k . a_{kj} . x_k . \gamma_{ikj} \quad \text{for } k \neq i \tag{6}$$

In equation (6), a_{kj} represents the propagation attenuation factor between node k and j and $\gamma_{ikj} \in [0,1]$ represents the probability that the packet transmitted by the interferer k would contribute to the interference at the receiver j. This factor is influenced by the medium access control selection. For example, for a CDMA system using a spreading factor F, such a probability is of about 1/F.

Robustness is defined as the probability that a message emitted at source S successfully arrives at the destination node D, and is referred to as $\boldsymbol{P}(R_{SD})$. Our aim is to maximize this probability. As energy and delay criteria need to be minimized, the robustness criterion is also formulated as a minimization criterion as: $f_R = 1 - \boldsymbol{P}(R_{SD})$.

$\mathbf{P}(R_{SD})$ can be defined as the probability that *the message arrives successfully in D in at most H hops, with* $H \to \infty$. Therefore, we have:

$$P(R_{SD}) = 1 - \prod_{h=1}^{\infty} (1 - P(R_{SD}|H = h)) \tag{7}$$

where $\boldsymbol{P}(R_{SD} \mid H=h)$ is the probability for a packet to arrive in h hops at the destination. We have the probability to reach the destination in 1 hop defined as $\boldsymbol{P}(R_{SD} \mid H=1) = p_{SD}$, the

successful transmission probability on the direct link between *S* and *D*. This probability is a function of the Bit Error Rate (BER) which depends on the Signal to Interference and Noise Ratio (SINR) of the link and on the transmission technology. When more hops are accounted for, $\boldsymbol{P}(R_{SD} \mid H=h)$ can be defined recursively as:

$$P(R_{SD}|H=h) = 1 - \prod_{j=1}^{N_S} [1 - p_{Sj} \cdot x_j \cdot P(R_{jD}\ H=h-1)] \tag{8}$$

where N_S is the number of possible first hop relay nodes of the source node S and p_{Sj} is the link probability between the source node and its neighbour *j*.

Since we cannot handle an infinite number of hops in the sum of Eq.(6), we need to set a maximum number of hops allowed in the communication: $H_{\max,}$. Once $H_{\max}$ is set, we will obtain the Pareto-optimal front for a delay-constrained network with a maximum allowed number of hops enforced. In this case, we can analyze the three-objective optimization problem knowing that the delay cannot exceed a value of $H_{\max}$. The computation of the sum for Eq.(6) can also be stopped when the $\boldsymbol{P}(R_{SD})$=1 (i.e. f_R=0). In this case, we have a two-objective problem which provides the trade-offs between delay and energy consumption when all the data is transmitted perfectly. There are solutions where $\boldsymbol{P}(R_{SD})$ can never reach 1 as there are not enough paths to the destination. In this case, such a solution is dropped from the neighbourhood.

The end-to-end transmission delay is given by the sum of the times spend at each relay node on a multi-hop path. Since in a fixed network propagation delays are negligible, the number of relays between source and destination is a good measure of the transmission delay. The criterion f_D is defined in this model as the 2nd order moment of the delay distribution among all the available paths:

$$f_D = \sum_{h=1}^{\infty} (h-1)^2 . R_h \tag{9}$$

The quantity (*h-1*) is the delay needed by a packet to arrive in *h* hops using (*h-1*) relay nodes. R_h is the probability that the packet arrived in *h* hops and did not arrive in 1, or 2… or (*h-1*) hops. For *h=1*, we have R_h=$\boldsymbol{P}(R_{SD} \mid h=1)$ and for *h*>1 we have:

$$R_h = P(R_{SD}|H=h) \cdot \prod_{i=1}^{h-1} (1 - P(R_{SD}\ H=i)) \tag{10}$$

The energy criterion f_E is defined as the average energy needed for a packet to reach its destination *D* starting from a source node *S*, whatever the path or number of hops needed. On a given path, the energy is the sum of the energy spent by all the hops that participate in the forwarding effort. We do not account for the energy needed by the source node to transmit its first packet. The average energy is the sum of the average energies needed to go from source to destination in *H = h* with at least one path. This criterion is defined as:

$$f_E = \sum_{h=1}^{\infty} E(R_{SD}|H = h) \tag{11}$$

where $E(R_{SD}|H = h)$ is the average energy needed for a successful transmission to D in h hops defined recursively as:

$$E(R_{SD}|H = h) = \sum_{j=1}^{N_S} p_{Sj} \cdot x_j \cdot e_j + E(R_{jD}|H = h - 1) \tag{12}$$

$E(R_{SD}|H = 1) = 0$ since the energy transmitted by the source node is not taken into account here.

Implementation of PMOTS

PMOTS has been adapted to address this combinatorial optimization problem. The same strategy as for the WLP problem in defining the search subsets is adopted here. Instead of gathering the solutions regarding the number of selected candidate APs, we gather them according to the number of forwarding sensors. In the same way, we can explore the subsets made of 1, 2, … or F forwarding nodes in the network. The size of each subset is given by the number of combinations of F elements out of N elements when a binary variable is considered (a sensor forwards or not).

The neighbourhood is here defined in the same way as for the WLP problem. There are swap, add and delete moves. Instead of applying these moves to the selected APs, we apply them to the forwarding nodes. Furthermore, instead of changing the direction of emission or the transmission power values, there is a probability of forwarding move where a new neighbour node sees its value of p_k being changed to another possible forwarding probability.

As for the WLP problem, solutions where numerous nodes forward at the same time are not profitable as they generate a lot of interference, reducing drastically the probability of correct reception while increasing inefficiently the energy consumption. Therefore, we also favor the search for solutions with a low number of forwarding nodes by choosing accordingly the starting solutions of the search front.

In the simple case where we only have one source-destination communication, it is clear that only a few relays are useful. When the number of concurrent communications increases, more sensors have to contribute to the forwarding effort. Therefore, for every problem instance, a good choice of the starting solutions is beneficial for the search.

In this chapter, we simply address the problem of a single source-destination transmission. A node density of ρ=0.7 is considered for a network consisting of N=334 nodes. For this particular problem instance, we set the maximum number of hops to H_{max}=4 and minimize robustness, delay and energy. There are 4 parallel search paths and the number of forwarding nodes is limited to 4. The initial front is composed of solutions with F=1, F=2, F=3 and F=4 solutions. The estimated Pareto front obtained after 1000 iterations is presented in Figure 12.

This Pareto front shows the trade-offs between robustness, delay and energy. A zero delay and energy is obtained for a reliability of 0.245. This particular solution reflects the direct source-destination communication where no other node is forwarding. For the solutions

with the highest values of energy, four relays actively participate in the broadcasting effort and achieve the best possible robustness. For all the other solutions, the number of broadcasting relays varies between 1 and 4 depending on the link quality.

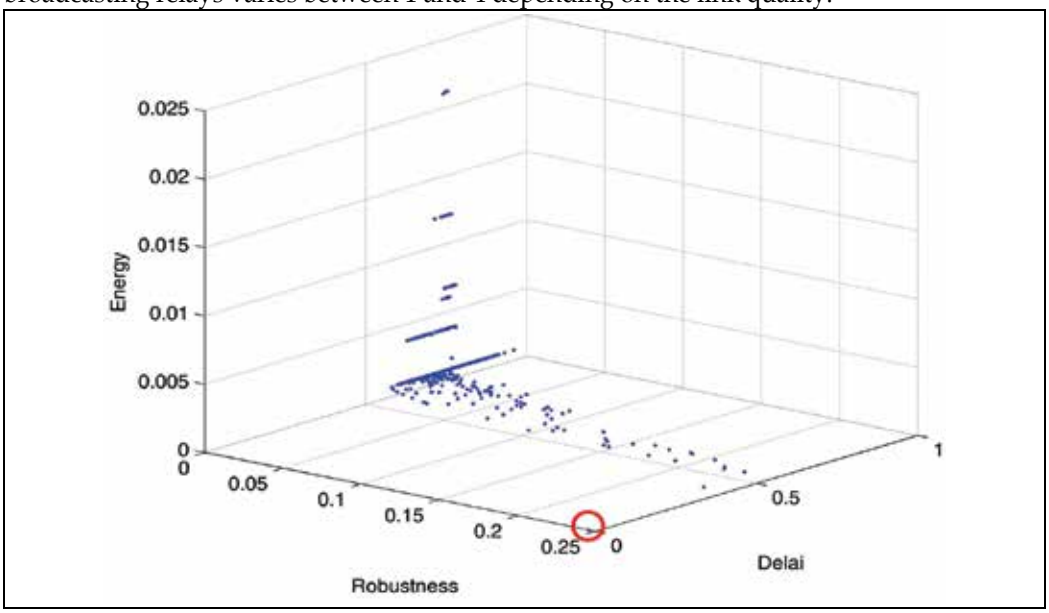


Fig.12. Estimated Pareto front for the single communication performance estimation problem

This representation of the Pareto front for the single communication problem is obtained after only about 635000 function evaluations. However, if more source destination pairs are considered in the network, solutions with a larger number of forwarding nodes are needed. In this case, the search can not simply target small parts of the solution space and a probably longer search would be necessary to investigate subsets with a higher number of forwarding nodes. As a consequence, we infer that the intensification strategy proposed at the end of section 5.1 will be very beneficial for the resolution of this problem as a more accurate estimate of the Pareto front is needed here.

6. Conclusion

As presented in this chapter, Tabu Search is a promising metaheuristic when addressing multiobjective optimization problems. It is particularly suited to handle problems with numerous combinatorial or continuous variables. The local search at its heart also makes it an interesting technique for highly constrained optimization problems. Most of the heuristics dealing with MO Tabu search use a single search path.

PMOTS, the algorithm used as an example in this work, relies on a simple parallel search which uses several paths and a specific neighbourhood construction strategy in order to spread the search paths in all the interesting parts of the solution space. The benefits of such an approach are highlighted for two particular problems arising in wireless systems. In the first one, a real world WLAN network has to be planned to meet several performance

guarantees. In the second one, a more theoretical use of the PMOTS algorithm is presented. In this particular case, the search really needs to provide the best possible estimate of the Pareto front for the results to be meaningful from the theoretical point of view. This contrasts with the WLAN planning problem where the search is asked to provide good solutions (but not optimal ones) in a limited search time. PMOTS performs very well for the WLP problem. However; the search time for convergence to the optimal Pareto front still needs some improvements. Therefore, an intensification stage that disregards bad performing paths and intensifies the search for promising ones might be beneficial to increase the convergence speed of the parallel approach.
From a more general point of view, multiobjective Tabu search is still a promising research area. The Tabu list at its core provides a mean to efficiently explore a good portion of the search space. Clever uses of Tabu lists that store information about the past solution evaluations have been proposed which improve the representation of the estimated Pareto front. Since Tabu search is easy to implement and performs nicely on a large set of single objective optimization problems, it makes it a good candidate for the fast implementation of a range of multiobjective real-world problems.

7. Acknowledgement

The work presented in this chapter has been supported by the Marie Curie program from the European Community's Sixth Framework Program. This chapter only reflects the Author's views and the European Community is not liable for any use that may be made of the information contained herein.

7. References

Aguado-Agelet, F.; Martinez, V.A.; Alvarez-Vazquez, L.J.; Hernando R.J.M. & Formella A. (2002). Optimization methods for optimal transmitter locations in a mobile wireless system. *IEEE Journal of Vehicular Technology,*Vol. 51, No. 6, 2002, 1316–21.

Armentano, V.A. & Arroyo, C.J.E. (2004). An application of a multi-objective Tabu search algorithm to a bicriteria flowshop problem. *Journal of Heuristics*, Vol. 10, No. 5, September 2004, 463–481

Bahri, A. & Chamberland, S. (2005). On the wireless local area network design problem with performance guarantees. *Computer Networks*, 2005,Vol. 48, No. 6, 856–66.

Baykasoglu, A.; Owen, S. & Gindy, N. (1999).A taboo search based approach to find the Pareto optimal set in multiple objective optimization. *Journal of Engineering Optimization*, Vol. 31, (1999), 731-748

Baykasoglu, A. (2006). Applying multiple objective Tabu search to continuous optimization problems with a simple neighbourhood strategy. *International Journal for Numerical Methods in Engineering*, Vol. 65, No. 3, 2006, 406-424

Collette, Y. & Siarry, P. (2004). *Multiobjective optimization: Principles and Case Studies.* Springer, ISBN 978-3-540-40182-7. Berlin.

Choobineh, F.F.; Mohebbi, E. & Khoo, H. (2006). A multi-objective Tabu search for a single-machine scheduling problem with sequence-dependent setup times, *European Journal of Operational Research*, Vol. 175, No. 1, (November 2006), 318-337

Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley, ISBN 047187339X, Chichester

Deb, K.; Pratap, A.; Agrawal, S. & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*. Vol. 6, No. 2, (April 2002), 182-197, ISSN: 1089-778X

De La Roche, G.; Jaffrès-Runser, K. & Gorce, J.-M. (2007). On predicting in-building WiFi coverage with a fast discrete approach. *International Journal of Mobile Network Design and Innovation 2007*; Vol. 2, No.1, 2007, 3 - 12.

Fonseca, C.M. & Fleming, P.J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*. Vol. 3, No. 1, (Spring 1995), 1-16, ISSN:1063-6560

Fonseca, C.M. & Fleming, P.J. (1998). Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms-Part I: A Unified Formulation. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 28, No. 1, 1998, 26—37

Gandibleux, X. & Freville, A. (2000). Tabu Search Based Procedure for Solving the 0-1 MultiObjective Knapsack Problem: The Two Objectives Case. *Journal of Heuristics*, Vol. 6 , No. 3, (August 2000) , 361 - 383, ISSN:1381-1231

Hansen, M.P. (2000). Tabu search for multiobjective combinatorial optimization: TAMOCO. *Control and Cybernetics*, Vol. 29, No. 3, 2000, 799–818

Ho, S.L.; Yang, S.Y.; Ni, G.Z. & Wong, H.C. (2002). A Tabu method to find the Pareto solutions of multiobjective optimal design problems in electromagnetics. *IEEE Transactions on Magnetics*, Vol. 38, No. 2, March 2002, 1013–1016

Jaeggi, D.M., Parks, G.T., Kipouros, T. & Clarkson, P.J. (2008). The development of a multi-objective Tabu search algorithm for continuous optimisation problems. *European Journal of Operations Research*, Vol. 185, March 2008, 1192-1212.

Jaffrès-Runser K.; Gorce, J.-M. & Ubeda, S. (2008). Mono- and multiobjective formulations for the indoor wireless LAN planning problem. Computers & Operations Research, Vol. 35, No. 12, December 2008, 3885-3901

Jones, D.F.; Mirrazavi, S.K. & Tamiz, M. (2002). Multi-objective meta-heuristics: An overview of the current state-of-the art, *European Journal of Operational Research*, Vol. 137, No. 1, February 2002, 1–9

Kulturel-Konak, S.; Smith, A.E. & Norman, B.A.(2006). Multi-objective Tabu search using a multinomial probability mass function, *European Journal of Operational Research*, Vol. 169, No. 3, (March 2006), 918-931

Reininger, P. & Caminada, A. (2001). Multicriteria Design Model for Cellular Network. Annals of Operations Research, Vol. 107, No. 1-4, October 2001, 251-265, ISSN 0254-5330 (Print) 1572-9338 (Online)

Sherali, H.D.; Pendyala, M. & Rappaport, T. (1996). Optimal location of transmitters for micro-cellular radio communication system design. *IEEE Journal on Selected Areas in Communications,*Vol. 14, No. 4, 1996, 662–73.

Srinivas, N. & Deb, K. (1994). Multiobjective Optimization Using Non dominated Sorting in Genetic Algorithms. *Evolutionary Computation.* Vol. 2, No. 3, 1994, 221-248

Van Veldhuizen, D.A. (1999). Multiobjective evolutionary algorithms: classifications, analyses and new innovations, Ph.D., Graduate School of Engineering. Air Force Institute of Technology, Wright Patterson AFB, Ohio, USA, January 1999

3

SOS-Heuristic for Intelligent Exploration of Search Space in CSOP

Jaziri Wassim
Miracl Laboratory, Higher Institute of Informatics and Multimedia, Sfax, Tunisia

1. Introduction

In optimization problems, the constraints play a fundamental role as restrictive clauses that limit the search process and the quality of solutions to be found.
To propose efficient solutions, it is necessary to satisfy stakeholders' constraints. However, stakeholders often require multiple and complex constraints, which are difficult to satisfy and constrain the task of optimization. In addition, some constraints require complex treatments, which are incompatible with a strategy of optimization and complicate the search for the best solution.
Indeed, in a constraint satisfaction process, a traditional problem is to verify if the set of constraints is coherent. As this problem is largely treated in literature, we consider that the set of constraints is coherent and the following problem consists in determining if the problem has a solution.
We deal with the framework of constraints satisfaction and optimization problems. An optimisation approach consists in exploring the search space to find the best solution according to an objective function [1][15]. This framework is then adapted to problems with multiple solutions. However, some real-life optimization applications are over-constrained and no solutions can be found. Therefore, the application of an optimization algorithm is not pertinent. The problem is how to act in order to face such a situation and how to take into account the constraints and their adequacy with the optimisation task?
Works about constraints satisfaction and optimisation problems often orient themselves toward the framework of optimisation or of the constraints satisfaction. We think that the two tasks of optimization and of constraints satisfaction must be treated simultaneously to find adequate solutions.
We seek in this work to develop a general approach of constraints satisfaction and optimization to ensure simultaneously the tasks of optimization and of constraints satisfaction. This approach is based on heuristics in order to guide the search for solutions. We illustrate our approach with examples from an application of optimization of the spatial distribution of crops in an agricultural territory.
This chapter is organized as follows. Section 2 presents an analysis of the problem and the need for an approach of optimization under constraints. We present in section 3 our satisfaction and optimization approach. Sections 4 and 5 develop a spatial optimization

problem related to the optimization of crops distribution on agricultural parcels as well as results of experiments in Seine Maritime, France. Section 6 presents an experimental model based on Tabu Search. We discuss and we conclude this work in section 7.

2. Constraint satisfaction or optimization process?

In real-life optimization applications, the treatments of users are in general simple and do not reflect the complex treatments necessary to the satisfaction of their constraints. The real complexity of constraints on the system level can penalize the search for optimal solutions. Then, the tasks of constraints satisfaction and optimization of the required objective can be incompatible. A solution largely used in the literature consists in reducing the problem by favoring the satisfaction of the most relevant constraints to the application according to an order of preference [6][7][18][19][20][22][23]. In addition, some methods have been developed, such as Branch and Bound [13]. This method uses heuristics to guide the search of good solutions. The solution quality depends on the treated problem and on the defined heuristics. Pre-processing techniques have also been developed to reduce the search space of constraints satisfaction problems and can be used in combination with other search techniques [4][12][16][17][21]. These methods allow us to avoid an enumeration of all potential solutions but they are very expensive in computation times.

2.1 Analysis of the search process

We analyse in this part the process of constraints satisfaction and of optimisation in order to understand the role of variables, by their respective values, in the improvement of the objective and in the satisfaction of the constraints. We thus propose an intelligent heuristic to orient the search of satisfactory solutions in constraints satisfaction and optimisation problems.

In practice, optimization problems can reach a high complexity and thus require high computing times because of the great number of potential solutions. Exact methods cannot generally treat this type of problem and the use of an approximate optimization method is the adequate way. However, an optimization process based on an approximate method requires a flexibility to explore the search space. It is generally not adapted to problems handling hard constraints.

Thus, the question is how to apply an optimization algorithm and to ensure at the same time the satisfaction of constraints, essential to propose satisfactory solutions to stakeholders? To provide a solution to this problem, we analyze the search process regarding separately the tasks of optimization and of constraints satisfaction.

An optimization approach affects values to *variables* in order to find the best combination of values which improves the objective function (cf., the neighbourhood methods [2]). The strategy of searching for solutions is based on a mechanism of neighbourhood exploration[1] guided by the evolution of the objective function.

A satisfaction approach affects values to *variables* in order to satisfy constraints on these variables. In this approach, the strategy of affectation is guided by the structure of the

[1] That depends on the applied algorithm.

constraints and their state of satisfaction during the resolution (cf., the backtrack methods [5]).
The processes of constraints satisfaction and of optimization are based on an affectation of values to *variables*, according to two different strategies. The variables have an important role since their respective values influence the quality of the solution according to the objective of optimisation and of constraints satisfaction. An interesting way consists in exploiting the role of variables to solve simultaneously the tasks of constraints satisfaction and optimization.

2.2 SOS-Heuristic : An intelligent heuristic to search solutions

A constraint can be defined as a condition to satisfy among a number of variables. It restricts the set of values that can be affected simultaneously to the implied variables [21]. A classic process of constraints satisfaction re-affects only the variables implied in constraints to satisfy. In the case of optimization, all variables likely to improve the objective value are re-affected, regardless of their implication in constraints. This re-affectation of variables can cause the dissatisfaction of some constraints that imply these variables. To remedy this problem, we define a heuristic 'SOS-Heuristic' (Heuristic for **S**atisfactory and **O**ptimized **S**olutions) to guide the re-affectation of variables with an intelligent manner. We thus choose to re-affect variables that don't satisfy constraints instead of re-affecting all variables in order to improve the objective function. The defined heuristic consists in affecting a penalty to each variable that violates a constraint. The variables that satisfy all constraints are not penalized. The penalized variables must be re-affected to satisfy the constraints which imply them. The objective of the SOS-Heuristic is to improve the value of the objective function without damaging the satisfaction of constraints.
Example. Let's consider the constraint of slope: "The parcels with a slope higher than 15° must not contain the corn".
All parcels with a slope higher than 15° are implied in this constraint. The implied parcels, which have a crop other than corn satisfy the constraint and are considered satisfactory. The implied parcels that are affected with the corn are unsatisfactory. To satisfy the constraint, we must affect the corn to all implied parcels. Thus, only the unsatisfactory parcels must be re-affected.
Moreover, in constraints satisfaction and optimisation problems, the constraints to satisfy do not generally have the same importance. Works about constraints satisfaction problems distinguish in particular hard constraints that should absolutely be satisfied and soft constraints that are to be satisfied as far as possible [20]. Thus, the value of penalty for each unsatisfactory variable can be adapted according to the level of preference of violated constraints (hard or soft).

3. The proposed approach

We consider here the framework of optimization while taking into account the violated constraints. We make abstraction of the optimization techniques and we not rely on a specific algorithm of resolution. However, we base our approach on the concepts of an approximate method that is compatible with big-size problems.

At the system initialization, a phase of constraints analysis identifies the variables which are implied in each one. The verification of values of these variables allows identifying the unsatisfactory variables[2]. The values of these variables must be modified during the optimization process in order to satisfy the constraints not yet satisfied. The search process remains oriented toward the improvement of the objective function but the re-affectation process is limited to the unsatisfactory variables.

3.1 Resolution strategy

The strategy of constraints satisfaction and optimization alternates a phase of optimization of the objective and a phase of satisfaction of the required constraints.

3.1.1 Optimization phase

The variables penalized during the initialization phase represent the input of the optimization phase: we limit the set of the optimisation variables to those violating constraints. These variables are re-affected during the optimization while encouraging the re-affectation of the most penalized variables[3]. In order to not deteriorate the initial satisfaction for the constraints, the variables that satisfy these constraints are not penalized and are not re-affected during the optimization. The re-affectation of penalized variables orients the search toward the satisfaction of the constraints initially violated and therefore the reduction of the number of penalized variables (which become satisfactory).

3.1.2 Satisfaction phase

After the optimization phase, the phase of satisfaction re-evaluates the satisfaction of constraints according to new values of the re-affected variables. This re-evaluation of the constraints satisfaction allows updating the set of unsatisfactory variables and thus the value of penalty for each one. The variables that have become satisfactory are removed from the initial set of unsatisfactory variables. The algorithm reiterates a new phase of optimization to improve the objective function by re-affecting a new set of penalized variables. This process is repeated until satisfying all constraints or until finding a good solution in relation with the value of the objective function and with the satisfaction of constraints.

In our approach, the variables have an important role in the exploration of solutions (during the optimisation phase) and in the evaluation of the constraints satisfaction (during the satisfaction phase).

3.2 Coding of the proposed approach

The proposed constraint satisfaction and optimization approach is based on a multi-phase mechanism that integrates the improvement of the objective at the level of the optimization phase and the satisfaction of the violated constraints at the level of the satisfaction phase. The algorithm is based on the following steps:

Algorithm 1:

---

[2] The variables that violate at least one constraint.

[3] The variables that violate several constraints or hard constraints.

Begin

/ Initialization phase */*

- Evaluate the initial satisfaction of the constraints

Extract the set of unsatisfactory variables /* the algorithm can re-affect only the variables that violate the hard constraints or can also re-affect all unsatisfactory variables */

- Attribute a penalty value for each unsatisfactory variable
- **Repeat** /* Do n simulations */

/ Optimization phase */*

- Apply an approximate optimization method /* the algorithm is independent of a specific optimization method */.
  - Re-affect the penalized variables
  - Evaluate the objective function

/ Satisfaction phase */*

- Evaluate the satisfaction of the constraints
- Update the penalized variables as well as their penalty value

- **Until** a stop condition /* a global satisfaction state */

End

Although it is guided by the evolution of the objective value, the exploration of solutions is based on the re-affectation of only the penalized variables. The choice of the variable to re-affect depends on the value of penalty in order to treat in first the most penalized variables.

4. Spatial optimization problem

The problem we are trying to solve consists in affecting crops on agricultural parcels in order de reduce runoff risks. Runoff risk is the consequence of the assignment of crops on the parcels. Runoff is the accumulation of outflows on the parcels, due to a weak infiltration of water in the soil. Above a certain rate of rain, the soil doesn't absorb water that flows according to the natural slope and causes losses of land (erosion) and a significant deterioration of the agricultural soil's surface. Important damage can be caused to the habitat and to the plantations, notably in the downstream areas [9][14].

To solve the runoff risk problem, we must optimize the distribution of crops on the parcels while respecting to the maximum the required constraints [10]. A configuration corresponds to a distribution of crops on the parcels. We evaluate the performance of each configuration according to the value of runoff risk and the average rate of satisfaction for the constraints. The satisfaction rate for each constraint is evaluated from the Database[4] using SQL (Structured Query Language) requests. The value of runoff risk is evaluated by a spatial hydrological model that calculates values of sensitivity to the runoff risk for each parcel. It also simulates the runoff risk due to a distribution of crops on all parcels [9].

4.1 A hybrid optimization method

As mentioned above, the developed approach is independent of a specific algorithm of resolution. To experiment our approach, we elaborated a hybrid method based on the evolutionary strategy [3] and the simulated annealing [11].

[4] The database is implemented in Relational Database Management System.

We consider the real distribution of crops already available, as a start configuration. We manipulate only one configuration (an individual in analogy with the evolutionary strategy) at a time.
From the initial configuration, the optimization process of runoff risk repeats a procedure of neighbourhood exploration based on the simulated annealing method. The mechanism of neighbourhood exploration is based on a permutation of crops between two parcels according to their sensitivity to the risk. The value of risk represents the function of cost. The objective consists in seeking configurations of weaker cost while accepting with a controlled manner the configurations that damage the function of cost. The new configuration is accepted systematically if the value of runoff risk decreases. Otherwise, the acceptance of a new configuration with a higher cost is determined with a probabilistic manner: a real $0 \leq \theta < 1$ is randomly chosen and then compared with a probability of acceptance $p(\Delta E, T)$. This probability is expressed according to the value of cost increase and the current temperature. The temperature is controlled by a decreasing function that defines a cooling schedule. The system starts with an elevated temperature that progressively tends toward 0 along the advancement in the process in order to lead the system toward a stable state. If $\theta \leq p(\Delta E, T)$, then the new configuration is accepted and replaces the current configuration. Otherwise, the current configuration is preserved and is reused to generate another one. The acceptance of increases allows us to leave the stable states of non-decrease of the risk. It also advantages the exploration of a large search space in order to avoid a premature convergence toward a local minimum. The probability of acceptance of risk increases is weak in order to avoid a random dispersal in the search space.

4.2 The satisfaction strategy

We based the resolution of the spatial optimization problem related to the runoff risk on the algorithm presented above (algorithm 1) and we favour the satisfaction of the hard constraints.
Indeed, the parcels can be implied simultaneously in hard and soft constraints. The treatment of all unsatisfactory parcels (that violate hard and soft constraints) can sometimes improve the satisfaction of the soft constraints at the expanse of the hard constraints. To avoid such situations, we have chosen to re-affect crops only to the parcels that violate hard constraints. We treat by permutations only the parcels whose crop doesn't satisfies at least a hard constraint. The parcels that satisfy all hard constraints remain invariant and maintain their initial crop during the optimisation phase.

4.3 The hybrid algorithm

The algorithm used to reduce the runoff risk is based on the following steps:
Algorithm 2:
Begin

- Start from the initial distribution $\mathbf{x_i}$ of crops on the parcels.

/ Initialization phase */*

- Evaluate the initial satisfaction of the constraints
- Extract the set **P** of parcels that violate hard constraints /* If all constraints are satisfied, extract the parcels that violate soft constraints and that are not implied in hard constraints */

/ Do n simulations */*

- **Repeat**
  / Optimization phase */*
  - Choose a high initial temperature **T** and a function α of temperature reduction.
  - **For** each parcel $\mathbf{p_i}$ from **P** ($\mathbf{p_i}$ is the most penalized parcel).
    - Generate a neighbourhood **x** (by permuting $\mathbf{p_i}$ with another penalized parcel).
    - If the risk value degrades then $\mathbf{x_i} = \mathbf{x}$.
    - Else, accept the new configuration with a random probability
    - Reduce the temperature $\mathbf{T} = \alpha\mathbf{(T)}$

  / Satisfaction phase */*
  - Evaluate the satisfaction rate of the constraints
  - Update the set **P** of penalized parcels.
  - Accept the solution if it satisfies the requirements of the application.
- **Until** a stop condition.

End

The algorithm seeks to reduce the risk value at the level of the optimization phase and to improve the satisfaction of hard constraints at the level of the satisfaction phase.

5. Experiments

We present in this part an experimentation that aims to reduce the runoff risk in the watershed of Haute-Durdent in Seine Maritime - France, while satisfying the constraints required by the farmers.

The watershed of Haute-Durdent (16 km²) has 450 parcels shared between a dozen of farms. Thirteen types of occupations: beet, wheat, wood, rape, fodder crops, escourgeon, fallow, linen, corn, potato, pea, prairie and village are placed on the parcels. The wood and village are considered as invariable and are not taken into account in the re-affectation process.

We consider 21 constraints: 16 hard and 5 soft, required by several farmers in their farms. These constraints express the requirements of slope, size and type of sol compatible with the placement of each crop as well as the production quantity of crops in each farm.

A solution to a constraints satisfaction and optimization problem is an affectation of values to the variables, which improves the objective value while satisfying the constraints. In our problem, we define a solution as a configuration that reduces the risk of runoff and maintains the initial rate of satisfaction for the hard constraints. The parcels are the variables of the system. Each parcel can be affected by a type of crop. The multiple possible affectations of crops on the parcels represent the search space.

5.1 Reduction of the risk value (the function of cost)

Several tests have been done in order to follow the evolution of the risk value (to minimize) during the optimization phase and at the end of each simulation. Fig. 1 shows the evolution of the risk value after each permutation during the optimization phase (inside the first simulation). Fig. 2 shows the evolution of the risk value after each simulation.

The results show a reduction of the risk value calculated by the spatial hydrological model at the outlet of the watershed. The developed algorithm converges after some simulations toward a stable state. Some deteriorations of the risk value (Fig. 1) are accepted during the optimisation phase in order to avoid local minima.

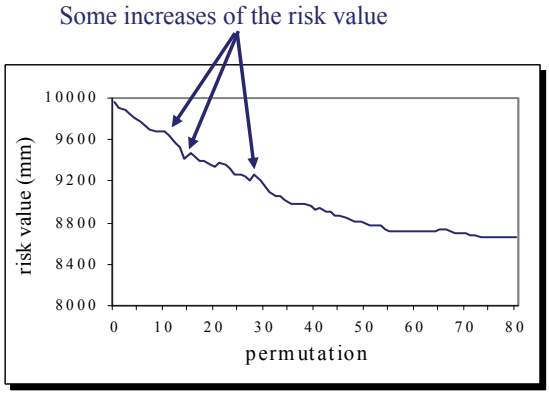


Fig. 1. The value of runoff risk after each permutation during the first simulation.

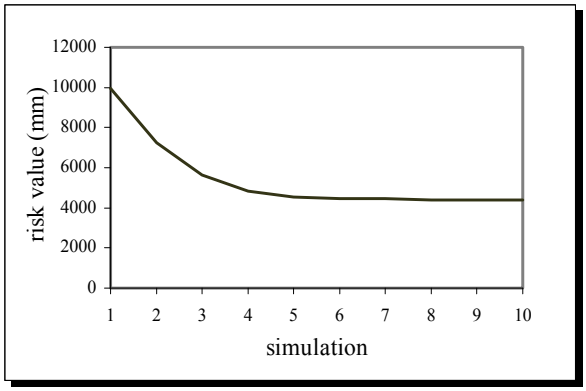


Fig. 2. The value of risk after simulations.

To evaluate the efficiency of the proposed algorithm and to explain the reduction of the risk value, we present in Fig. 3 the number of permutations and of the penalized parcels during simulations.

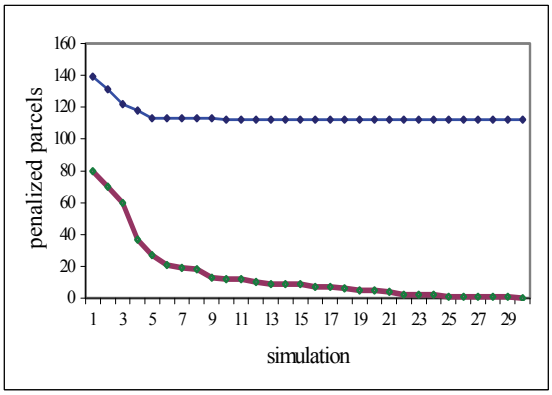


Fig. 3. The number of permutations (thick feature) and of penalized parcels (thin feature) through simulations.

The number of crops permutations between the parcels tends toward 0 and explains the convergence of the algorithm after some simulations toward a stable state. The number of penalized parcels decreases by 19,42% (from 139 to 112) after 10 simulations (Fig. 3).
Although the number of treated parcels is restricted since we treated only the parcels that violate hard constraints, the value of risk decreased by 56% after 10 simulations (from 9967 to 4377 mm, Fig. 2). This reduction obtained by the treatment of 30% of the parcels (139 among 450 parcels, Fig. 3) proves that the aggravation of the risk is essentially due to elementary parcels that are badly managed by farmers. Consequently, the affectation of appropriate values to variables which cause the dissatisfaction of constraints allows solving efficiently an optimization and constraints satisfaction problem.

5.2 Evaluation of the constraints satisfaction

We evaluate the average rate of satisfaction for the hard and the soft constraints (Fig. 4). The treated variables can be implied simultaneously in several constraints. A complex constraint that implies a big number of variables cannot often be satisfied by all implied variables and only a subset of variables satisfies the constraint. To this effect, we consider that a constraint is satisfied with a certain degree[5] rather than satisfied (100%) or dissatisfied (0%). The satisfaction of a constraint is evaluated according to the ratio: the number of satisfactory variables / the total number of variables implied in the constraint. For example, the satisfaction of the constraint of slope defined above is evaluated according to the ratio: the number of satisfactory parcels (that have a slope higher than 15° and contain a crop different from corn) / the total number of parcels implied in the constraint (that have a slope higher than 15°).

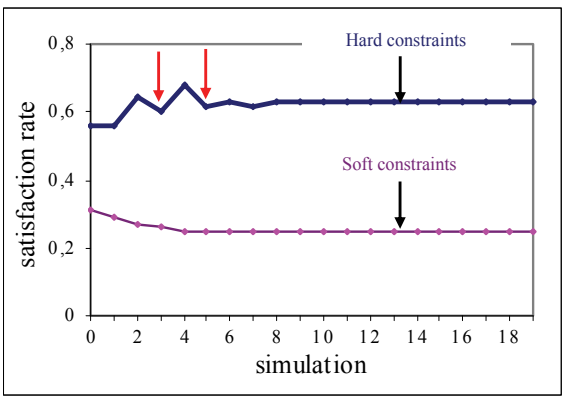


Fig. 4. The average rate of satisfaction for the constraints through simulations.

The defined SOS-Heuristic allowed us to maintain (and improve) the initial rate of satisfaction for the hard constraints (Fig. 4). However, we note some deteriorations of the satisfaction rate without descending beyond the initial rate of satisfaction (at the simulation 3 and 5, Fig. 4). Indeed, at each simulation, the algorithm accepts a deterioration of the satisfaction rate to overtake local maxima.

[5] We find the principles of the Fuzzy Constraints Satisfaction Problems [7].

5.3 Spatial consistency

We visualize on the following maps the initial distribution of crops on the parcels of Haute-Durdent (Table 1 and Fig. 5) as well as the distribution proposed by the developed approach (Fig. 6).

Colour assigned for each type of crop

Watershed of Haute-Durdent	Type of crop	Colour
	beet	
	wheat	
	wood	
	rape	
	fodder crops	
	escourgeon	
	fallow	
	linen	
	corn	
	potato	
	pea	
	prairie	
	village	

System of simulation

Table 1. The colour assigned for each type of crop is darker as the crop is pro-runoff.

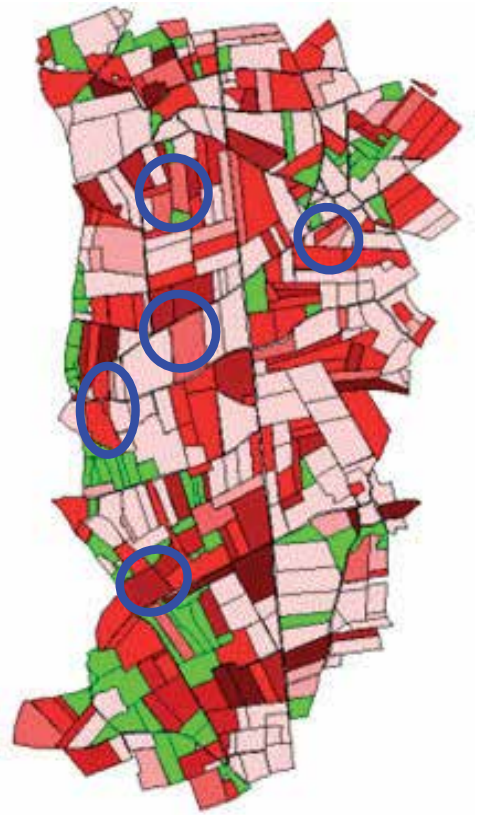


Fig. 5. The initial distribution of crops on the parcels of Haute-Durdent.

Fig. 6. The distribution of crops proposed by the developed approach.

The optimization process dissociates the regrouping of pro-runoff crops in the sensitive zones to minimize the risk (Fig. 5 and 6, circled parts).
The high complexity of the required constraints as well as the few number of treated parcels (we treated only 30% of the parcels) explain the small, but efficient, number of modifications made on the initial distribution of crops.

6. An experimental model based on Tabu search

To better analyse the contribution of the SOS-Heuristic, we present in this part an experimental optimization model based on Tabu search. This experimentation aims to redistribute crops between the parcels of Haute-Durdent without taking into account all stakeholders' constraints. Thus, we do not use the SOS-Heuristic and our primary objective is to reduce the runoff risk.

6.1 Tabu search optimization

To optimize the distribution of crops on the parcels, we used an approximate method based on Tabu search [8]. Tabu search is an optimization metaheuristic, belonging to the class of local search techniques [1]. A local search technique is an iterative process based on two essential elements: a neighbourhood and a procedure exploiting this neighbourhood.

Starting from an initial configuration (solution), a typical local search method explores the search space to replace the current configuration by one of its neighbours in order to minimize a function of cost (or to maximize a function of benefit).
We consider the real distribution of crops as an initial configuration. We manipulate one configuration at a time. A configuration is a distribution of crops on all parcels of the territory. The process of crops permutation between two parcels acts as an explorer of the search space. It generates, after each permutation, a neighbouring configuration that minimizes the risk value. From the initial configuration, the optimization process permutes the crop of the most sensitive parcel with that of another parcel in the territory. The optimization algorithm doesn't choose the first permutation that improves (minimizes) the risk value. All possible permutations are enumerated to discover the most profitable configurations. This solution is certainly expensive, but the discovered neighbouring configuration will be of a smaller risk value. The configuration generated after a permutation is accepted systematically if the run-off risk decreases. Otherwise[6], the last least expensive permutation is chosen, according to a probability of acceptance, in order to escape the stable states of non-reduction of risk. The acceptance of increases of the risk value encourages the exploration of a vast search space in order to avoid a premature convergence of the optimization algorithm.
However, to prevent cycles, the last explored configurations are kept in a short-term memory (Tabu list). This list memorizes information about the history of the last generated configurations in order to prohibit permutations that lead to one of them. It thus allows the algorithm to explore other parts of the search space. In addition, to avoid a memorization of expensive configurations, we memorize only the permutations that generate them.

6.2 Optimization strategy

The optimization algorithm is based on the following steps:
Algorithm 3:
Begin
While (Untreated parcels exist) do
1. *Start from an initial empty Tabu list*
2. *Select p, the most sensitive of the untreated parcels.*
3. *Select C, the set of candidate parcels that can exchange their crops with that of p while satisfying the constraints.*
4. *Eliminate parcels that take permutations already explored.*
5. *Classify the retained parcels according to their contribution to the runoff risk.*
6. *Permute the most sensitive parcel with another parcel.*
7. *Update the Tabu list*

End While.
End
The treatment of a parcel consists in searching a permutation of its crop with that of another less sensitive in the watershed, in order to minimize the risk value. A simulation consists in

[6] When no permutation minimizes the risk after some permutations.

treating all parcels of the watershed by searching for possible permutations among the occupying crops.

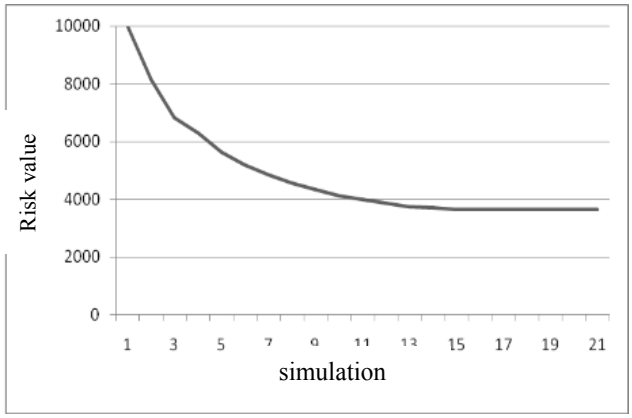


Fig. 7. The value of risk after simulations.

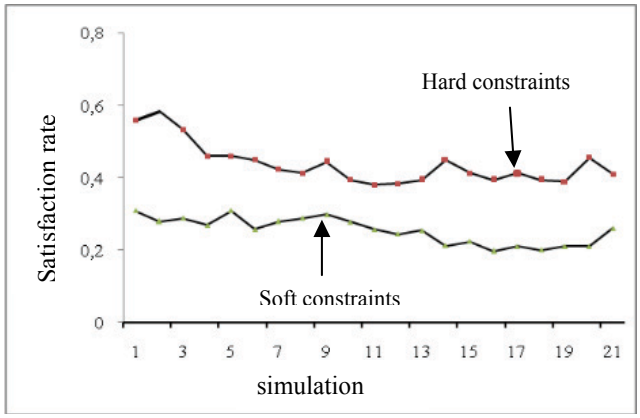


Fig. 8. The average rate of satisfaction for the constraints through simulations.

The experimental model has reduced the value of the risk calculated at the watershed outlet (Fig. 7). The runoff value decreased by 63% after 20 simulations (from 9967 to 3640 mm, Fig. 7). This reduction is obtained by the treatment of all 450 parcels. However, only some restricted constraints are taken into account.

In addition, we represent on the following map the distribution of crops proposed by the experimental optimization model (Fig. 9).

As shown in Fig. 9, to reduce the risk, the experimental optimization model re-distributes crops randomly since it does not take into account the spatial constraints related to the placement of crops. Consequently, the distribution of crops proposed by this model is not adapted to the reality of land and does not respect farmers' requirements:

- The satisfaction of the production objectives for crops is not satisfied: farmers require that a quantity of production must be reached for each type of crop. This quantity is expressed in this problem by a total surface to cover by each crop in the territory.
- The spatial distribution of crops on the parcels is incoherent and the map obtained in Fig. 9 is not accepted by farmers.

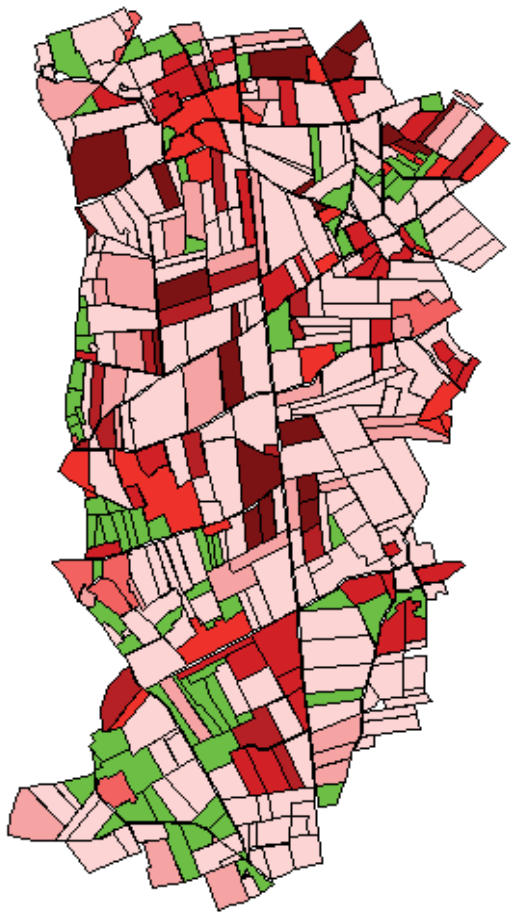


Fig. 9. The distribution of crops proposed by the optimization model.

7. Discussion and conclusion

We have proposed in this work a constraints satisfaction and optimization approach to orient the search process in optimization problems. We have developed a generic approach that allows to solve simultaneously the task of optimization and of constraints satisfaction based on an intelligent search heuristic : the SOS-Heuristic. The experiments concern a spatial optimization problem related to a distribution of crops on agricultural parcels to minimize the runoff risk. The application of a hybrid method based on the simulated annealing and the evolutionary strategy provided interesting results and proves the efficiency of the proposed approach regarding the minimization of the risk value as well as the satisfaction of the farmers' constraints.

We have also presented an experimental model based on Tabu Search. This model, modelised into the framework of combinatorial optimization, is not oriented toward the constraints satisfaction since no global analysis has been conducted at this level. Only a limited set of constraints, like the requirements of slope and type of soil, is taken into

account. The satisfaction of these constraints is explicitly provided by eliminating the assignments that not verify them in order to limit the search space[7]. The simplified model of the problem allowed a certain degree of flexibility to the system thanks to its little number of constraints. This explains the multitude of changes which occured on the distribution of crops in the territory. However, this resolution is not realistic since it does not take into account all farmers' constraints and it considers only some soft constraints. The solutions proposed by this experimental model are not accepted by the farmers. Thus, the optimization and satisfaction approach based on the SOS-Heuristic is more efficient than the experimental model and provide more realistic solutions. It could be used to solve other constraints satisfaction and optimization problems.

8. References

E. Aarts and Lenstra J.K., Local search in combinatorial optimisation, Wiley, Chichester, England, 1997.

R.K. Ahuja, Ö. Ergun, J.B. Orlin and A.P. Punnen, A survey of very large-scale neighbourhood search techniques, Discrete Applied Mathematics, volume 123, pp. 75-102, Elsevier Science Publishers, Amsterdam, 2002.

T. Bäck, Evolutionary algorithms in theory and practice, Oxford University Press, USA, 1996.

R. Dechter and I. Meiri, Experimental evaluation of pre-processing techniques in constraint-satisfaction problems, Proceedings of the 11th International Joint Conference on Artificial Intelligence, pages 290-296, USA, 1989.

V. Dechter and D. Frost, Backjump-based backtracking for constraint satisfaction problems, Artificial Intelligence, pp. 147-188, 2002.

D. Dubois and P. Fortemps, Selecting preferred solutions in the minimax approach to dynamic programming problems under flexible constraints, European Journal of Operational Research, pp. 582-598, 2005.

H. Fargier, J. Lang and T. Schiex, Selecting preferred solutions in fuzzy constraint satisfaction problems, Proceedings of the 1st European Congress on Fuzzy and Intelligent Technologies (EUFIT), pp. 128-134, Allemagne, 1993.

F. Glover and M. Laguna, Tabu Search, Kluwer Academic Publishers, 1997.

W. Jaziri and T. Paquet, A Multi-Agent Model and Tabu Search Optimization to Manage Agricultural Territories, *GeoInformatica- An International Journal on Advances of Computer Science for GIS*, volume 10, Issue 3, pages 337-357, Springer, Kluwer Academic Publishers, USA, 2006.

W. Jaziri, Multi-Scale Optimization for the Management of Run-off Risks in Agricultural Watersheds, *IEEE Transaction on Systems, Man and Cybernetics*, Part C : Applications and Reviews, volume 37, Issue 4, pages 573-582, 2007.

C. Koulamas, S.R. Anthony and R. Jean, A survey of simulated annealing: application to operations research problems, Omega, pages 41-56, 1994.

[7] We find the concepts of pre-processing techniques.

V. Kumar, Algorithms for constraint satisfaction problems: a survey, Artificial Intelligence, volume 13, pp. 32-44, American Association for Artificial Intelligence, USA, 1992.

E.L. Lawler and D.E. Wood, Branch and bound methods: a survey, Operations Research, volume 14, pp. 699-719, 1996.

P. Martin, Reducing flood risk from sediment-laden agricultural runoff using intercrop management techniques in northern France, Soil and Tillage Research, 52: 233-245, 1999.

C.H. Papadimitriou and K. Steiglitz, Combinatorial optimization: Algorithms and complexity, Prentice Hall, 1982.

P. Prosser, Hybrid algorithms for the constraint-satisfaction problem, Technical Report, AISL-46-91, Department of Computer Science, Université de Strathclyde, Scotland, 1991.

P. Prosser, MAC- CBJ: Maintaining arc-consistency with conflict-directed backjumping, Technical Report 95/177, Department of Computer Science, Université de Strathclyde, Scotland, 1995.

T. Schiex, Possibilistic constraint satisfaction problems or "How to handle soft constraints?", Proceedings of the 8th International Conference on Uncertainty in Artificial Intelligence (UAI-92), pp. 268-275, USA, 1992.

T. Schiex, H. Fargier and G. Verfaillie, Valued constraint satisfaction problems: Hard and easy problems, Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95), pages 631-637, 1995.

A.S. Tanguiane, Aggregation and representation of preferences, Springer-Verlag, Berlin, 1991.

E.P.K. Tsang, Foundations of constraint satisfaction, Academic press, London, England, 1993.

A. Tversky, Intransitivity of preferences, Psychological Review, pages 31-48, 1969.

P. Vincke, Aggregation of preferences: A review, European Journal of Operational Research, pp. 17-22, Elsevier, 1982.

4

Symbiotic Tabu Search

Ramin Halavati[1] and Saeed Bagheri Shouraki[2]
[1]*Iranian Academic Centre for Education, Culture, & Research*
[2]*Sharif University of Technology*
Iran

1. Introduction

Since the introduction of Tabu Search (Glover, 1989 & 1990), (Glover & Laguna, 1997), this simple yet effective heuristic has been used in many applications such as clustering (Liu et al., 2008), rule base generation (Bagis, 2007), feature selection (Oduntan et al., 2008), multi objective optimization (Zhu et al., 2007), (Vilcot, 2007), (Jaeggi et al., 2008), combinatorial optimization (Crainic et al., 2007), (Alvarez-Valdes et al., 2006), continuous optimization (Jaeggi et al., 2008), scheduling (Pan et al., 2007), (Chen et al., 2006), solving graph problems (Mermri et al., 2007), (Öncan et al., 2007), (Brandão & Eglese, 2006), and many more.

Tabu search metaheuristic has been used in two forms by now. The first form is the pure usage of tabu metaheuristic as the search algorithm. In this approach, the algorithm designer just focuses on implementing appropriate neighbour generation functions based on her/his specific task. Examples of this approach are (Pacheco et al., 2007) for urban transport optimization, (Liang & Chao, 2008) for facility layout optimization, (Exler et al., 2007) for control and system design, (Palubeckis, 2007) for maximum diversity problem, (Caserta & Uribe, 2007) for software system reliability, (Watcharasitthiwat et al., 2006) for design of FIR filters, and (Hallberg & Peng, 1996) for multicycle scheduling.

The second approach is to hybridize this idea with another search algorithm or another instance(s) of the same idea. Examples are hybridization with Memetic algorithms (El Fallahi, 2006), Particle Swarm Optimization (Shen et al. 2007), Genetic algorithms (Hou et al., 1999), (Vilcot & Billaut, 2007), and used in multiple/hierarchical form in (Oduntan et al., 2008), (Brandão, 2007), (Crainic et al., 2007), (Pothiya et al., 2007), (Watcharasitthiwat et al., 2006) or in parallel (Al-Yamani et al., 2003), (Bachelet et al., 1996), (Porto & Ribeiro, 1996).

Cited hybridizations can be categorized into two sub groups: either tabu search is the master search algorithm which makes use of another algorithm as slave for neighbour generation, as in (Hou et al., 1999), or the other algorithm is the master algorithm which uses tabu search for either initialization (Vilcot & Billaut, 2007) or local optimization (Gomes et al., 2003). In this chapter, we want to purpose another form of hybrid tabu search approach in which the tabu metaheuristic is neither the master part of the algorithm nor its slave part and two ingredients, tabu metaheuristic and symbiotic evolution, are intermixed so that none has a higher position than the other. The algorithm will be called Symbiotic Tabu Search (STS) and will be a general search and optimization algorithm that makes use of schemata theorem (Holland, 1975), automatically discovers and uses the linkages between solution parts and avoids local optima.

In the rest of the chapter, we will first introduce the rationales behind symbiotic tabu search and why we were led to this idea. Then an implementation will be presented in section 3, followed by its comparison and test results in section 4, and finally the concluding remarks at section 5.

2. The backgrounds

In this section, we will first talk about the linkage problem, which is the main focus of Symbiotic Tabu Search algorithm, then will introduce symbiotic combination operator as an artificial tool inspired from the natural process of symbiogenesis. Then in the next section, we will present STS algorithm as the combination of symbiogenesis and tabu search as a remedy for the linkage problem.

2.1 The linkage problem

One of the earliest ideas in search algorithms was to use the building blocks of a problem (divide and conquer) and to combine the building blocks so that the good features of two partially good solutions would be added together and create a better solution. This was the main idea behind the recombination (cross over) operator of Holland's Genetic Algorithms (Holland, 1975) which be came the root of all evolutionary algorithms. But very soon three problems arouse in this field which were laterly called Linkage Problem: First, how to identify the good building blocks so that their combination will result in a good fitness value (Watson & Pollack, 1999); how to combine these blocks if they are conflicting (Watson & Pollack, 1999); and how to identify and what to do with bad genes which are stuck to good genes in one chromosome, the garbage genes (Forrest & Mitchell, 1993). The first problem results in requirment of prior domain knowledge for positioning genes so that recombination opertor would not break the building blocks, and the second and third problem results in slowing down, and sometime shutting down, the search process by approving and propagating bad genes.

These two problems result in a vast line of search methods based on building blocks or schemata theory with several different view points. The first ideas were based on designing more sophisticated recombination operators for simple genetic algorithms such as the ones with more number of cut points, random cut point positioning, uniform crossover, linear combination of genes, etc., see (Mitchell, 1999) for an extensive list. Regardless of the fact that some of these remedies totally neglect the idea behind schemata and building blocks, prior domain knowledge about the problem is still a serious requirement for selection/design of appropriate operator.

Another idea was to use chromosome reordering operators and repositioning of genes inside the chromosome on the fly such as Inversion operator (Bagley, 1967) and Linkage Learning Genetic Algorithm (Harrik, 1997). In such algorithms, each gene has a location indicator along with its value and both of these parameters change and propagate during evolution. Thus, gene values and locations are optimized together and the algorithm is supposed to rearrange gene locations so that those which have related effects on phenotype move together during application of simple recombination operators. The major reported problem of this approach was premature convergence or reaching a local optima before the genes were appropriately arranged (Newman, 2006), (Pelikan et al, 1999).

The third solution was to use partially specified chromosomes (PSCs) such as in Messy Genetic Algorithms (mGA) (Deb, 1991), (Goldberg et al, 1989), Cooperative Co-Evolutionary

Algorithms (CCEA) (Potter & De Jong, 1994), Symbiotic Evolutionary Adaptation Model (SEAM) (Watson & Pollack, 2000) and, and Incremental Commitment Genetic Algorithm (ICGA) (Watson & Pollack, 1999). In these approaches, the chromosomes have missing values for some locations and cooperation of several chromosomes (based on algorithm strategy) composes a solution. In mGA, this cooperation is minimal and each PSC is evaluated in the context of a template and is compared with other chromosomes which are similar enough to it. In CCEA, genome is split into several sub-genomes by algorithm designer and each pool is evolved in separation from other pools, improving the content of one of these sub-genomes. The cooperation between pools takes place during evaluations were a partial solution from one pool is concatenated to the bests of the other pools for evaluation. In SEAM and ICGA, PSCs are evaluated in the context of other chromosomes. In this approach, a context is a combination of some members of the chromosome pool that fully specify all chromosome locations together. To compute the fitness of a chromosome, all unspecified positions of the chromosome are filled with respective values from the the context and then the fitness value is computed.
The last school of dealing with linkage and building block problems tries to find estimation of distribution of good genes. In contrast with purely evolutionary approaches that put their direct focus on search for good solutions, the algorithms in this group try to estimate the distribution of good genes and construct good solutions based on these estimations. Generally, these approaches require a pre-selected distribution model and the process estimates the parameters of this distribution. To name some algorithms in this category, one can say Estimation of Distribution Algorithm (Larrañaga & Lozano, 2002), Population-Based Incremental Learning Algorithm (Baluja, 1994), Compact Genetic Algorithm (Harik et al, 1998), Extended Compact Genetic Algorithm (Sastry & Goldberg, 2000), Factorized Distribution Algorithm (Mühlenbein & Mahnig, 1999), Bayesian Optimization Algorithm (Pelikan et al, 1999), and Hierarchical Bayesian Optimization Algorithm (Pelikan et al, 2003). Each of these solutions has its own cons and pros and yet none is considered an ultimate remedy for the linkage problem. Some of the approaches require other source of domain knowledge like estimation of distribution of good genes in all fourth approach algorithms and appropriate gene groups in CCEA; Some need excessive computation power for extensive search as in mGA, see more details in (Kargupta, 1995); and at last, some are usable only in very specific purposes such as SEAM, see more details in (Halavati et al, 2007).

2.2 Natural process of symbiogenesis and artificial symbiotic combination operator

The natural process of symbiogenesis (Merezhkovsky, 1909) is the creation of new species from the genetic integration of organisms, called symbionts. Symbiogenesis has enabled some of the major transitions in evolution (Maynard Smith & Szathmary, 1995), including the origin of eukaryotes which include all plants and animals. This kind of genetic integration is quite different from the transfer of genetic information in sexual reproduction. Sexual recombination occurs between similar organisms (i.e. of the same species) and involves the exchange of parts of the genome in a mutually exclusive manner so that every gene acquired from one parent is a gene that cannot be acquired from the other parent. In contrast, symbiotic combination may also occur between genetically unrelated organisms (i.e. different species) and involve the integration of whole genomes. The resultant composite may have all the genes from one symbiont and at the same time acquire any number of genes from the other symbiont (Watson & Pollack, 2000).

Based on this idea, symbiotic combination operator was introduced (Watson & Pollack, 1999), (Watson & Pollack, 2000) as an alternative for sexual recombination operator. This operator takes two PSCs and makes an offspring with the sum of their characteristics, see Figure 1 for an example. Therefore, in contrast to the standard crossover operator that is applied to fully specified chromosomes, symbiotic combination runs over partially specified representations and advances them towards fully specified ones. In the original introduction of this operator ((Watson & Pollack, 1999) and (Watson & Pollack, 2000)), when two chromosomes with conflicting genes were to be merged (i.e. they both had values for one/more specific location and the values contradicted), all these conflicts were resolved to the favour of the first donor. In this text, we do not need this assumption and we simply don't combine two chromosomes that have conflicts.

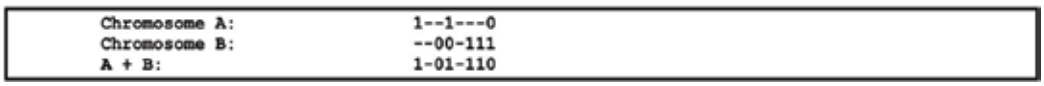


Fig 1. An example of symbiotic combination. Chromosomes A and B, each, have some unspecified locations, shown with '-' mark. Their combination has specified values for all locations that are specified in at least one of the donors. If there would be a conflict between the specified values, like the last gene of the above chromosomes, all conflicts are resolved in favor of one donor, here A.

3. Symbiotic tabu search

3.1 The idea behind symbiotic tabu search

The main target of STS is to use partially specied chromosomes (PSCs) as the core of the search approach, but make the selections using tabu prohibition heuristic on fully specified solutions. To do so, STS uses mutation operators, sometimes combine PSCs using symbiotic combination operator to create chromosomes with more specified locations, and makes the selection for reproduction using tabu heuristic and fitness value of fully specified solutions.

One requirement of a process that evolves PSCs is their evaluation during selection phase. In some specific problems, this is done using a direct evaluation function that can evaluate solutions with missing values; but when this is not available, the most common approaches are to evaluate a partially specified chromsome in the context of other members of the pool such as in SEAM and ICGA or to use a template for missing values as in mGA.

In the first approach, a context is a combination of some members of the pool that fully specify all chromosome locations. To compute the fitness of a chromosome in a context, all unspecified positions of the chromosome are filled with respective values from the context and then fitness value is computed. A major critique of this approach is the detachment between the context and the chromosome: When the combination of several chromosomes (the context) and the chromosome under evaluation results in a good fitness value, it means that the entire group has made a good cooperation and their being together results in a good outcome. So it would be good if the entire composition would get a higher chance of reemergence and survival, but in this class of algorithms, only the single individual under evaluation gets the reward, see (Halavati et al, 2007) for more details on this problem.

The local template solution of mGA also has this problem as selection is done at individual level. Also the process is limited to a local search based on the template that is used for

evaluations and although the template evolves through time, all evaluations are centered around its temporal value in each generation.
To overcome these problems, we propose evaluation of chromosomes in the context of other chromosomes, but the reproduction and selection must also be done at context level instead of individual level: if some chromosomes that are grouped for evaluation gain a high fitness value, all of them get a higher survival and reproduction chance and not just one of them. To implement this idea, we will use the term *assembly* hence forth for a set of none conflicting chromosomes that fully specify all locations; i.e., each location is specified by one and only one member of the assembly. Each member of an assembly will be called a *symbiont*. This way, whenever we need selection or evaluation, we can generate an assembly, evaluate it and if it gains enough credits for reproduction, reproduce its entire content. But after replication (and possible modifications) it can break again into its symbionts.
There are some considerations behind the recommended process:

3.1.1 Avoiding premature convergence

Creating schemata with more specified bits from the best assemblies in each iteration may result in a very fast creation of fully specified chromosomes and premature convergence of the search process. To prevent this, we limit the size of chromosomes that are created during the process to a value that gradually increases while the process goes on. This value will allow the creation of only single gene chromosomes at the beginning of the process and gradually reaches fully specified chromosomes. This will be called gradually cooling the process and the term is chosen to represent a force that prevents emergence of big chromosomes at the beginning of the process, when the pools is hot, and gradually cools the pool based on a predefined schedule, so that bigger chromosomes can emerge. In all of our implementations, this parameter is simply computed by a linear equation based on the generation number but more complicated functions can be used if the problem space is known better.

3.1.2 Avoiding local optima

STS algorithm creates schemata of the best assembly in each iteration by combining some symbionts of the best assembly. This is done to promote solutions similar to the best assembly, but if the best assembly is a local maximum, there would be a good chance that exactly the same assembly will be chosen again as the best assembly of the later steps, because all symbionts of this solution still exist in the pool and some combinations of them are also added. In this case, the local maximum will fill up the chromosome pool very fast with more and more copies of its subcomponents and this increases the possibility of its later creations and may cause the search to get stuck there. The key role of tabu prohibition is in this step where STS keeps a list of the latest best assemblies that have reproduced and avoids them during later assembly creations. This way, a local optimum can not repeat itself and this prohibition makes the algorithm search around the local maximum instead of repeating the exact assembly.

3.1.3 Exploiting partial evaluation function

If a partial evaluation function exists, STS can use it: Whenever such a function exists, STS can use it during assembly generation phase, by picking the first member of the assembly

randomly and then choosing the next members using a tournament selection approach. If the problem does not permit partial chromosome evaluation, this part can be replaced with a random selection of members for assembly build up.

3.1.4 Pruning none fitting chromosomes

Besides promoting and replicating good chromosomes, STS prohibits and decreases the reemergence of bad assemblies. To do so, removal of some chromosomes of assemblies with low fitness values is a good solution, but this must be done considering the fact that some of these chromosomes may also be in assemblies with high fitness values and they must be preserved. So, if a chromosome is part of both high fitness assemblies and low fitness assemblies, it must not be removed.

3.2 The implementation details of symbiotic tabu search

Figures 3 and 4 depict the implementation of STS. Figure 3 presents the main body of the algorithm and Figure 4 shows the assembly generation function which is in charge of creating assemblies when required. Assembly generation function (Figure 4) starts with a random chromosome and keeps adding chromosomes to the assembly so that new members have no conflicts with previous members and add some extra specified bits to the assembly. Two alternatives are shown in the diagram, once a partial evaluation function does not exist, new members are randomly selected from candidate members and when there is such a function, the best of each group of candidates is selected and added.

In the main body of the algorithm (Figure 3), as stated in 3.1.1 subsection and to avoid premature convergence, a size control mechanism is implemented using *MaxSize* variable that is initialized to 2 bit chromosomes at the beginning and is updated at the end of each iteration. A tabu list is also created in step 1 of Figure 3 and everytime an assembly is selected as the best of an iteration, re-creation of it in some further steps is prohibited by putting it in a queue of tabu answers in step 4. Note that assemblies are checked for being tabu in the final stage of assembly generation function and if an assembly with similar values for all locations of the generated assembly was found in tabu list, the newly generated assembly is discarded.

As stated in steps 3, 5, and 6 of Figure 3 diagram, only the best assembly of each generation is selected for mutation and symbiotic combination. In mutation step (5), with a certain probability, a mutated copy of each symbiont of the best assembly is created and added to the pool and in symbiotic combination step (6), combinations of each two symbionts of best assembly are created and added to the pool. At the end of step 6, it is checked that if the size of a combined symbiont exceeds maximum size threshold, it is randomly broken into some fragments of smaller sizes.

After replicating the best assembly, we prohibit the re-emergence of the worst assemblies in step 7 similar to the note at subsection 3.1.4: We separate the generated assemblies into two sets. The winners list includes all symbionts of all assemblies that stand in 25% highest ranks based on their fitness values and the losers list is made up of all symbionts of the worst 25% assemblies. Then, all symbionts of all members of the losers set are removed from the population, except the ones which are also a member of the winners set. Being in both sets is quite possible and frequent as the assembly generation phase may use a chromosome in

several assemblies with different fitness values. Using the above approach, we only remove chromosomes which haven't been able to take part in any good assembly.
After all stated steps, if the replication phase has created any duplicate chromosome in the population, the extra copies are removed in step 8 and if population exceeds a pre-specified threshold, some chromosomes are randomly selected and removed from the pool.

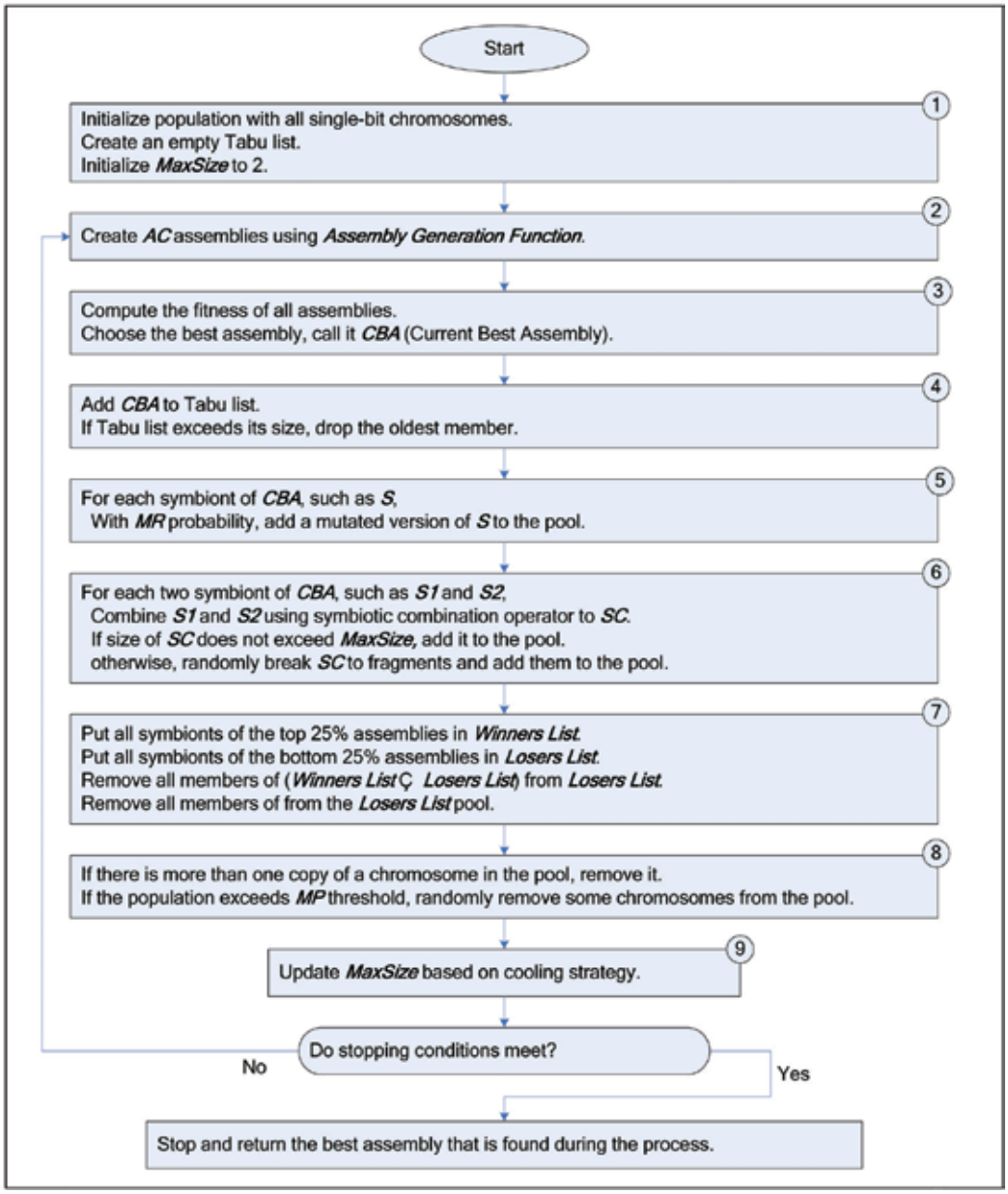


Fig. 2. Diagram of STS algorithm for optimization. The parameters are AC for Assemblies Count, MR for Mutation Rate, and MP for Maximum Population threshold.

3.3 Feature comparison of STS with some other algorithms

As stated before, there are generally three methods to deal with linkage or building block problems. One method is the estimation of gene distribution algorithms such as EDA (Larraaga & Lozano, 2002), PBIL (Baluja, 1994), cGA (Harik et al, 1998), ECGA (Sastry & Goldberg, 2000), FDA (Mühlenbein & Mahnig, 1999), BOA (Pelikan et al, 1999), and hBOA (Pelikan et al, 2003). As STS does not lie in this category, its features can not be easily compared with these algorithms because the cited algorithms require a previously selected distribution model for good genes while STS does not need it; but if such model exists, STS can not make use of it.

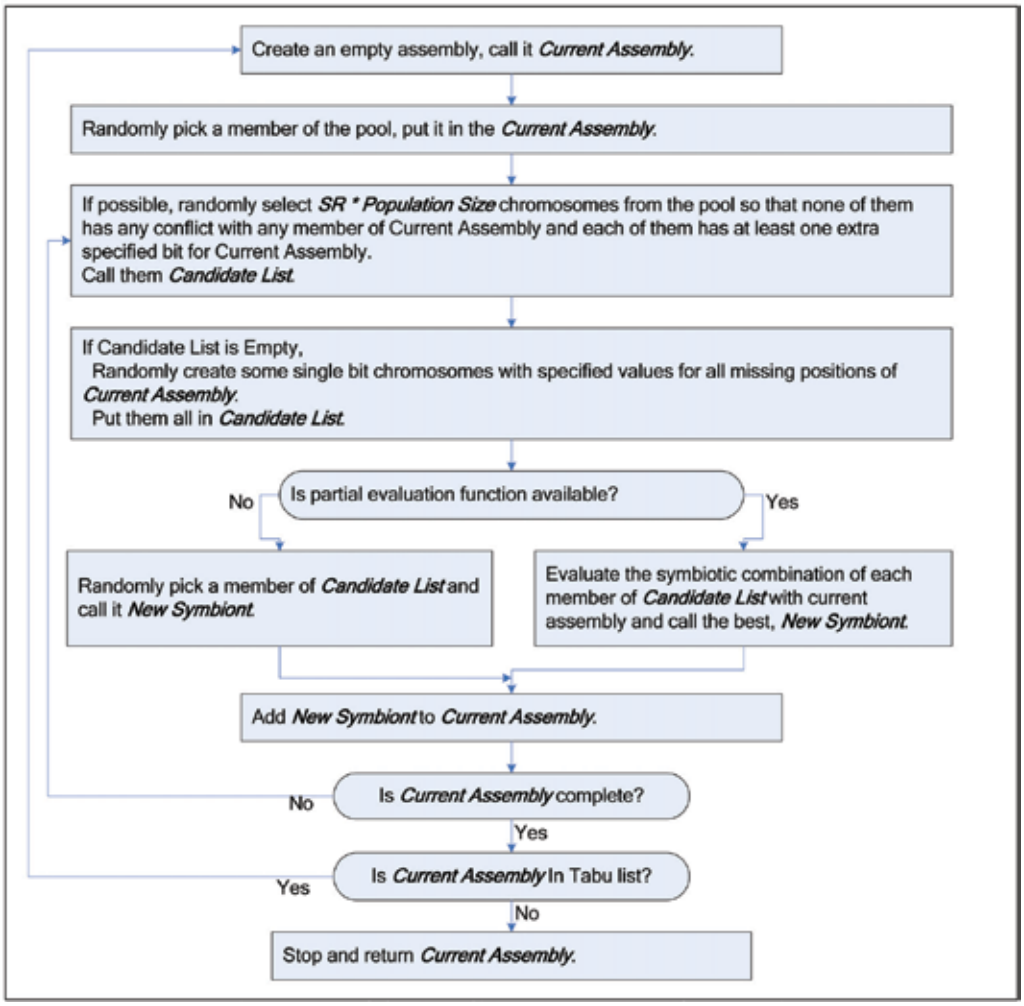


Fig. 3. Diagram of Assemly Generation Function. The parameters are *SR* for Selection Rate and *Population Size* for the number of chromosomes in the pool.

In Table 1, we have compared STS with some major algorithms of *partial chromosome specification* and *chromosome reordering* schools. As stated there, STS does not use fully specified chromosomes, so does not have the garbage genes problem; it doesn't work under the bias of any conflict resolution method as it does not combine conflicting chromosomes; it

is not under the influence of size penalty function as it lets the chromosomes grow only if they prove have high fitness value in cooperation with each other; it requires no prior domain knowledge or partial evaluation function, but it can make use of both if available; it uses substructures and can gradually evolve and reform them and is not bound to algorithm designer for substructure definition.

	SGA	mGA	SEAM	CGA	LLGA	CCEA	STS
Fully Specified Chromosomes?	Yes	No	No	No	Yes	No	No
Fixed Chromosome Locations?	Yes	Yes	Yes	Yes	No	Yes	Yes
Sensitive to Genome Order?	Yes	No	No	No	No	No	No
Carries Garbage?	Yes	No	No	No	Yes	No	No
Conflict Resolution Method?	Random	In Favor of First Parent	In Favor of First Parent	Most Fit Parent	Remove Originals	Not Needed	Not Needed
Diversity Preservation Method?	Mutation	Thresholding	Deterministic Crowding	Mutation	Mutation	Mutation	Mutation
Requires Size Penalty Function?	No	No	No	Yes	No	No	No
Requires Domain Knowledge?	Yes	Yes	Yes	Yes	No	Yes	No
Uses Substructures?	No	Yes	Yes	Yes	No	Yes	Yes
Substructures Evolve?	No	Yes	Yes	Yes	No	No	Yes

Table 1. Feature Comparison of STS and some other algorithms

4. Experimental results

4.1 Benchmarks problems

We used three benchmark problem sets. The first one is the Hierarchical If and Only I (HIFF) function (Watson & Pollack, 1999) with fully deceptive behaviour. The function takes an N-bit input and computes the fitness as stated in equation (1).

$$F(B)=\begin{cases}1, & \text{if } |B|=1\\ |B|+F(B_L)+F(B_R), & \text{if } (|B|>1) \text{ and } (\forall i,\{b_i=0\} \text{ or } \forall i,\{b_i=1\})\\ F(B_L)+F(B_R), & \text{otherwise}\end{cases} \quad (1)$$

B_L and B_R are respectively the left and the right half of B bit string.

The second benchmark is the concatenation of multiple 8-Queen problems (M8Q) (Eiben et al, 1995). In each instance, M separate problems of putting 8 queens on an 8×8 chessboard must be solved, so that no two queens on a board can attack each other. The chromosome includes the rows of the queens and columns are all assumed distinct and fixed.

The third benchmark is the KN-Trap function (Kargupta, 1995). The chromosomes are concatenations of K sections of N bits. For each N bit section, the fitness is computed as stated in equation (2) and the summation of all fitness values is assigned to the whole chromosome. The Trap function has a deceptive behaviour with the global maximum for an all-zero bit string and a negative gradient towards this point.

The chromosomes in all problems are shuffled (Watson & Pollack, 1990), so that adjacency data may not be used by any of the algorithms.

$$F(B) = \begin{cases} \text{SizeOf(B)}, & \text{if } Ones(B) = 0 \\ \text{Ones(B) -1}, & Otherwise \end{cases} \quad (2)$$

Ones(B) equals to the number of bits in B with value 1.

4.2 Benchmarks algorithms

We compared STS with Hill Climbing (HC) (Russle and Norvig, 2002), Symbiotic Evolutionary Adaptation Model (SEAM), simple Genetic Algorithm (SGA), and plain Tabu Search algorithm (TSA). HC was chosen as the most primitive local search algorithm, SEAM from symbiotic algorithms family, TSA from tabu search family, and SGA as the basic evolutionary algorithm.

To select appropriate parameters for SEAM, TSA, SGA, and STS on each instance of the benchmark problems, we used a greedy optimization algorithm with 100 steps and 20 random restarts. The measures for selecting appropriate parameters where first, number of solved problems, and second, the time needed to solve the problems. The best parameter settings, for each algorithm/problem instance is specified in Tables 2-5. For each algorithm, problems which could not be solved with any set of parameters are marked with N/A for not available.

Each instance of the benchmark problems was tried with each algorithm in 30 independent runs. Each algorithm/problem was given a maximum number of allowed fitness function calls based on problem complexity. These maximum values were chosen based on our initial experiments and were the same for all algorithms (10 times the fastest result that we found on that problem).

The cooling function of STS is implemented as a linear function, by dividing the iteration number per *CoolingRate* parameter. For each algorithm/problem, the success rates (number of times that each algorithm could find the optimum value (Forrest & Mitchell, 1993b)) and the average number of fitness computations for cases in which each algorithm has been able to solve the problem are depicted in respective diagrams.

	HIFF-5	HIFF-6	HIFF-7	1x4 Trap	4x4 Trap	2x8 Trap	1x8 Queens	5x8 Queens	10x8 Queens	15 x 8 Queens
Number of Contexts	50	100	150	50	50	100	50	N/A	N/A	N/A

Table 2. Parameters of benchmark problems for SEAM algorithm.

	HIFF-5	HIFF-6	HIFF-7	1x4 Trap	4x4 Trap	2x8 Trap	1x8 Queens	5x8 Queens	10x8 Queens	15 x 8 Queens
Tabu List Size	100	N/A	N/A	100	100	100	N/A	100	N/A	N/A

Table 3. Parameters of benchmark problems for TSA algorithm.

	HIFF-5	HIFF-6	HIFF-7	1x4 Trap	4x4 Trap	2x8 Trap	1x8 Queens	5x8 Queens	10x8 Queens	15 x 8 Queens
Population Size	300	N/A	N/A	100	100	N/A	100	100	100	500
Mutation Rate	0.75	N/A	N/A	0.65	0.65	N/A	0.7	0.65	0.95	0.75
Cross Over Rate	0.7	N/A	N/A	0.7	0.65	N/A	0.6	0.65	0.5	0.75
Elitism Percentage	0.1	N/A	N/A	0.1	0.3	N/A	0.15	0.3	0.1	0.3
Selection Method[1]	RW	N/A	N/A	TS	RW	N/A	TS	RW	TS	RW

Table 4. Parameters of benchmark problems for SGA Algorithm

	HIFF-5	HIFF-6	HIFF-7	1x4 Trap	4x4 Trap	2x8 Trap	1x8 Queens	5x8 Queens	10x8 Queens	15 x 8 Queens
Number of Assemblies	25	50	75	25	25	50	25	100	75	90
Cooling Rate x 1000	15	17.5	20	10	12.5	15	11	12.5	12.5	20
Selection Rate	0.001	0.001	0.001	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Mutation Rate	0.5	0.5	0.5	0.7	0.7	0.6	1	1	1	1
Number of Tabus	200	200	200	100	100	100	100	100	100	100

Table 5. Parameters of benchmark problems for STS Algorithm.

4.3 Experimental results of HIFF problems

The success rates and performance of the five tested algorithms on HIFF problems are depicted in Figure 5 diagrams. As it is seen, the only algorithm which was not able to solve all instances of 32 bit HIFF was SGA; TSA and HC totally failed to solve the 64 bit instances of HIFF while SGA still solved 20% of instances of 64 bit HIFF; and on 128 bit problems, only 30% were solved by SEAM while STS succeeded in 95% of cases.

Also it must be noted that in 32 bit problems, TSA and HC used almost double th computation time that STS required to solve the problem and generally, STS required much more computation time in compare with HC an TSA.

This problem has a fully deceptive behaviour, and identification and using building blocks is a serious requirement of the task. That's why SGA failed from the beginning. HC and TSA where successful at the smallest case where the search space was still not very large and found the solutions with checking almost $1/10000^{th}$ of all possible solutions, but with bigger search spaces they totally failed to find the solution by plain tracing and without using schemata. The only two algorithms that survive till 128 bit problems are STS and SEAM while STS is ahead with 65% more success rate, but exploiting much more computation time, even in smaller problems. This is because STS takes much more time analysing different cases (using cooling mechanism) before it composes a bigger building block but SEAM makes building blocks much faster and in the first success of a building block. Therefore SEAM can converge much faster than STS, but as it is seen, in big problems where there are much more local optima this may lead to a wrong turn to a local pick.

[1] RW for Rolette Wheel / TS for Tournoment Selection

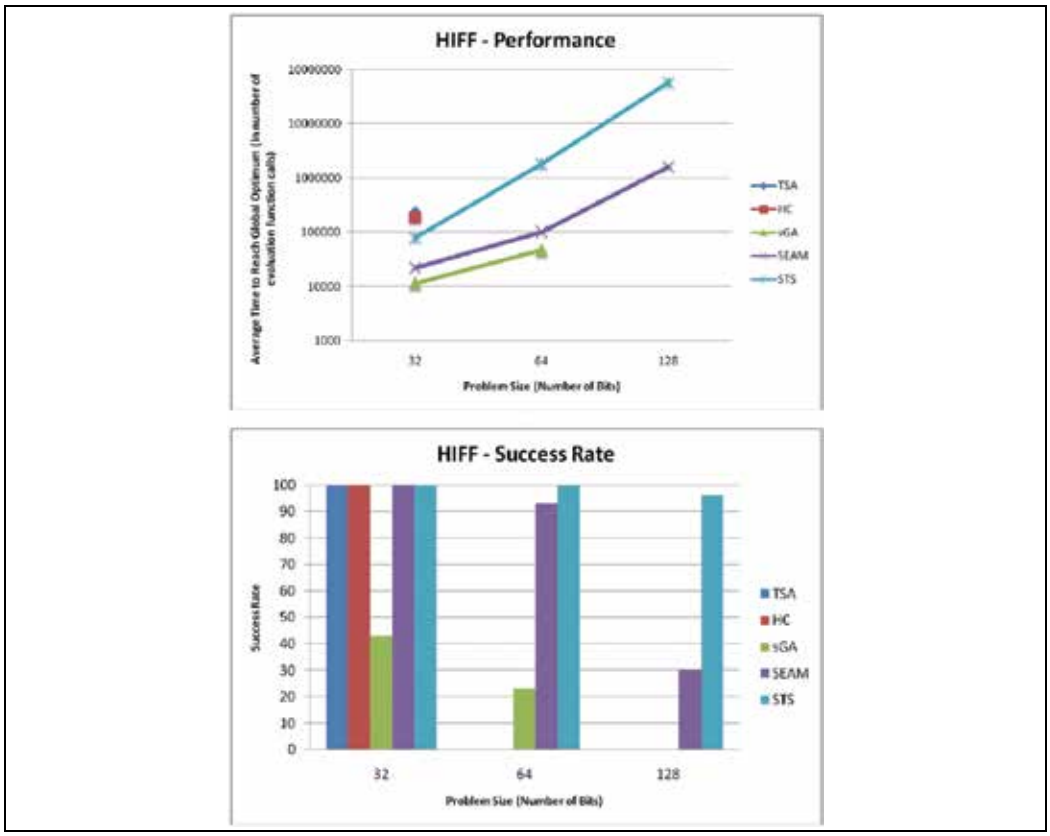


Fig. 5. Performance and Success rates of HC, TSA, SGA, SEAM, and STS algorithms for HIFF Problems.

4.4 Experimental results of M8Q problems

As depicted in Figure 6, the smallest instance of M8Q problem is solved completely by all algorithms, but the only ones that keep solving the bigger instances are SGA and STS. The reason behind the failure of HC and TSA can be in the big search space of 5 and more number of boards (120 bit for 5 boards, 240 bits for 10 boards and 360 bits for 15 boards) and the excessive number of local optima of M8Q problems.

SEAM also failed soon due to its nature of partial chromosome evaluation which creates a bigger chromosome, only if it gains better fitness value in compare with both its parents, in the context of many other chromosomes. This evaluation fails in M8Q problem as there are many global optima whose components are not a good solution of other global optima and therefore, the combination of each two chromosomes may be a good subcomponent of one solution and a bad subcomponent for another solution.

The only algorithms that could solve the 5, 10, and 15 board problems where SGA and STS and as depicted in Figure 6, the computation time of STS was somewhat lower than SGA and the success rate, more than 50% better in the biggest problem. Again we believe that the better success rate of STS in compare with SGA is due to its ability to create appropriate building blocks and therefore optimization of different boards almost separately, while SGA can not do this due it its fixed structured recombination operators.

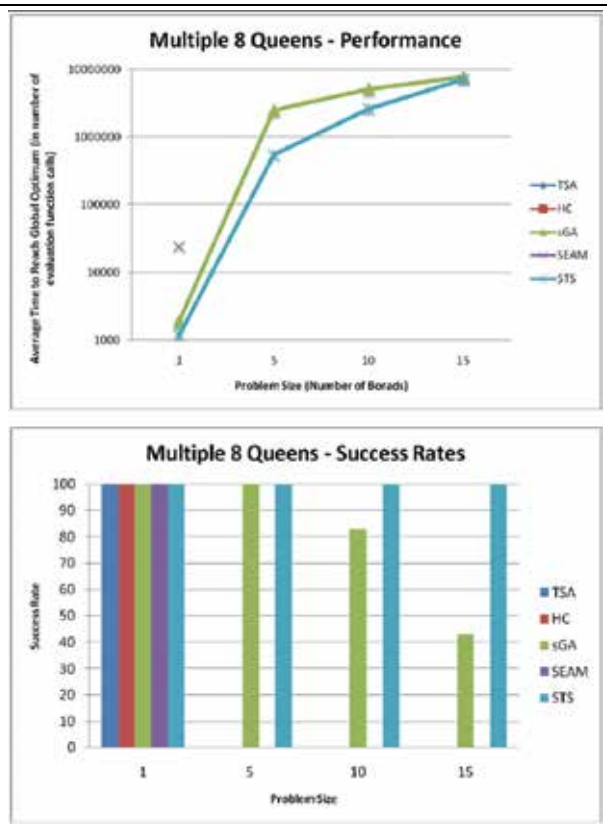


Fig. 6. Performance and Success rates of HC, TSA, SGA, SEAM, and STS algorithms for Multiple 8 Queens Problems.

4.5 Experimental results of KnTrap problem

The results of KnTrap problems are presented in Figure 7. This problem has a relatively very small size (16 bits in the biggest instance), but due to its deceptive behaviour and positive gradient only towards local optima, it is a very hard problem for greedy algorithms. HC and TSA were quite successful in all instances of this problem, as the search space is very small and due to the internal restart and choosing new random start point of both algorithms, they find the global optimum after enough number of restarts. SGA fails in 4×4 and 2×8 instance of this problem as the selection force always leads it to local optima and distances it from the best answer, and SEAM fails for the same reason on the biggest instance. But STS still keeps solving all instances, with more computation time in compare with HC and TSA, as its cooling mechanisms tries to find building blocks of each size, and not just based on previously found building blocks. So it ultimately finds partial optimum solutions of the size of trap functions and by combining them, finds the global solution.

5. Concluding remarks

To overcome the linkage problem, we proposed the usage of partially specified chromosomes (PSCs) along with evaluation and selection at group level. The main point

behind this idea was to select and replicate PSCs at group level instead of individual level, because once a group of PSCs show a good fitness value together, it does not implicate that each of them is a good subsolution in general, but it means that they are good together and can form a good cooperation; So, once we evaluate a group and they show good fitness value, all members of the group receive a higher change for survival and replication. During replication, to increase the chance of regrouping for some members of a successful group, we stick some of them together using symbiotic combination operator and build partial solutions with more specified bits. If these partial solutions show good fitness values in their future combinations with other members of the pool, they would be replicated again and grow bigger and if not, they will be destroyed. As this template is quite prone to getting stuck in local optima, it is augmented with tabu prohibition mechanism to avoid this problem and perform a better global search. The major advantages behind this idea are:

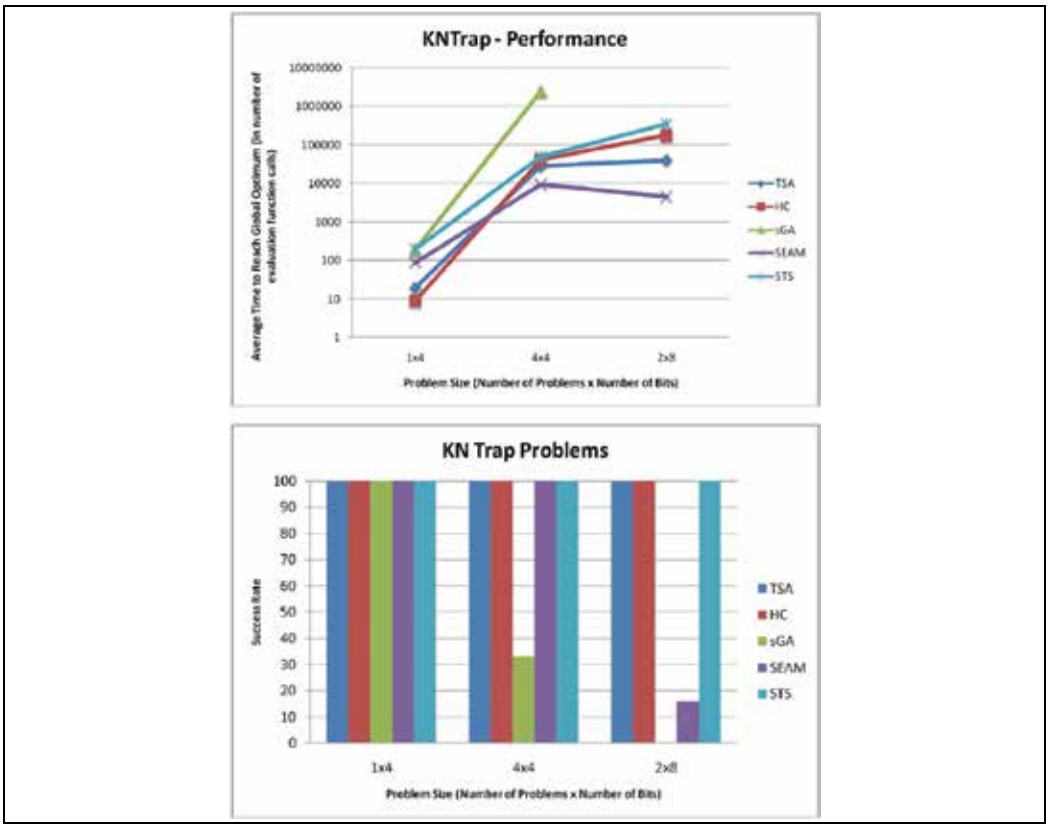


Fig. 7. Performance and Success rates of HC, TSA, SGA, SEAM, and STS algorithms for KnTrap Problems.

1. The evolutionary algorithm does not need prior domain knowledge for chromosome or recombination operator design for preservation of building blocks during recombination, as there is no recombination any more.
2. The process uses schemata and sub structures during evolution and can find and use building blocks, while the building blocks have no predefined structure or size

limitations. Moreover, growth control is done only based on fitness and therefore there would be no bias due to size penalty function or anything similar.

3. As the process uses PSCs and creates bigger chromosomes only if the group of chromosomes shows a high fitness value, garbage genes are produced less than approaches based on fully specified chromosomes.
4. Tabu prohibition prevents getting stuck in local optima and Cooling mechanism ensures a wide global search before narrowing down to areas limited by schemata.

The Symbiotic Tabu Search (STS) algorithm was presented in section 3 and was compared with hill climbing (HC), plain tabu search algorithm (TSA), simple genetic algorithm (SGA), and symbiotic evolutionary adaptation model (SEAM) in section 4 on three families of benchmark problems. The problems were chosen from 3 categories, a fully deceptive problem (HIFF), a combinatorial optimization problem (M8Q), and concatenation of some deceptive functions (KnTrap).

Our test results totally complied with our assertions about STS and showed that STS was quite more successful in reaching optimum solutions using less computational power, except in cases that search for building blocks was useless and solutions were quite trivial. In these cases, the gadgets of STS were just unnecessary and only time consuming, but yet STS could find the optimal solution with more computations.

As a final statement, we believe that evaluation and selection at group level can be a major advantage to search and optimization methods that are based on partially specified chromosomes and symbiotic combination is an appropriate tool for building schemata in such algorithms. Along with these two, tabu prohibition is a mandatory mechanism to prevent local optima as this search strategy is quite prune to getting stuck in the first local optima. Putting these along the facts that this algorithm does not require prior domain knowledge for operator or chromosome design, does not require any sort of partial evaluation function or size limit functions but can use them if available, and can create building blocks with no pre-specified limit, STS can be introduced and recommended as a general search and optimization algorithm that can automatically discover and use building blocks to cope with any form of internal relation between different parts of the solution.

6. Acknowledgements

Authors wish to give their sincerest thanks to Professor Caro Lucas for his valuable comments during this task, and Ms. Mojdeh Jalali Heravi & Ms. Bahareh Jafari Jashmi for their help through implementation and tests.

7. References

Al-Yamani, A.; Sait, S.M.; Barada, H. & Youssef, H. (2003). Parallel tabu search in a heterogeneous environment, *Proceedings of Parallel and Distributed Processing Symposium,* 8 pp. on CDROM, ISBN: 0-7695-1926-1, Nice, France, 22-26 April 2003.

Alvarez-Valdes, R.; Parreño, F. & Tamarit, J.M. (2006). A tabu search algorithm for a twodimensional non-guillotine cutting problem. *European Journal of Operational Research,* Vol. 183, No. 3, 16 December 2007, 1167-1182.

Bachelet, V.; Preux, P. & Talbi E-G. (1996). Parallel Hybrid Meta-Heuristics: Application to the Quadratic Assignment Problem. *Proceedings of the Parallel Optimization Colloquium*, pp. 233-242, Versailles, France, March 1996.

Bagis, A. (2007) Fuzzy rule base design using tabu search algorithm for nonlinear system modeling, *ISA Transactions,* Vol. 47, No. 1, January 2008, 32-44.

Bagley, J.D. (1967). *The Behaviour of Adaptive Systems Which Employ Genetic and Correlation Algorithms*, PhD Dissertation, University of Michigan.

Baluja, S. (1994). *Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning*, Tech. Rep. No. CMU-CS-94-163. Pittsburgh, PA, Carnegie Mellon University.

Brandão, J. (2007). A deterministic tabu search algorithm for the fleet size and mix vehicle routing problem, *European Journal of Operational Research,* in Press, available via Science Direct, doi:10.1016/j.ejor.2007.05.059.

Brandão, J. & Eglese, R. (2006). A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers & Operations Research,* Vol. 35, No. 4, April 2008, 1112- 1126.

Caserta, M. & Uribe, A.M. (2007). Tabu search-based metaheuristic algorithm for software system reliability problems, *Computers & Operations Research*, in Press, available via Science Direct, doi:10.1016/j.cor.2007.10.028

Chen, L.; Bostel, N.; Dejax, P.; Cai, J. & Xi, L. (2006). A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. *European Journal of Operational Research*, Vol. 181, No. 1, 16 August 2007, 40-58.

Crainic, T.G.; Perboli, G. & Tadei, R. (2007). TS_2PACK: A two-level tabu search for the threedimensional bin packing problem, *European Journal of Operational Research,* in Press, available via Science Direct, doi:10.1016/j.ejor.2007.06.063.

Deb, K. (1991). *Binary and floating point function optimization using messy genetic algorithms* (IlliGAL Report No. 91004). Urbana: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Eiben, A.E; Raué, P.E. & Ruttkay, Z. (1995). *GA-easy and GA-hard Constraint Satisfaction Problems*. In: Constraint Processing, Manfred Meyer (Ed.), Springer-Verlag LNCS 923, 267-283.

El Fallahi, A.; Prins, C. & Calvo, R.W. (2006). A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem, *Computers & Operations Research,* Vol. 35, No. 5, May 2008, 1725-1741.

Exler, O.; Antelo, L.T.;. Egea, J.A.; Alonso, A.A. & Banga, J.R. (2007), A Tabu search –based algorithm for mixed-integer nonlinear problems and its application to integrated process and control system design, *Computers and Chemical Engineering,* in Press, available via Science Direct, doi:10.1016/j.compchemeng.2007.10.008.

Forrest, S. & Mitchell, M. (1993). Relative Building-block fitness and the Building-block Hypothesis, *In Foundations of Genetic Algorithms 2*, L. D. Whitley (Ed.), 109-126. Morgan Kaufmann, San Mateo, CA.

Glover, F. & Laguna, M. (1997). Tabu Search. Kluwer Kluwer Academic Publishers, Boston.

Glover, F. (1989). Tabu Search, Part I, *ORSA Journal on Computing,* 1, 190-206.

Glover, F. (1990). Tabu Search, Part II, *ORSA Journal on Computing*, 2, 4-32.

Goldberg, D.E.; Korb, B. & Deb, K. (1989) Messy Genetic Algorithms: Motivation, analysis, and first results. *Computer Systems*, 3, 5, 493-530.

Gomes, L. C. T.; de Sousa, J. S. ; Bezerra, G. B.; de Castro, L. N. & Von Zuben, F. J. (2003) Copt-aiNet and the gene ordering problem, *Proceedings of Second Brazilian Workshop on Bioinformatics*, Rio de Janiro, Brazil, 3-5 December 2003.

Halavati, R.; Shouraki, S.B.; Jashmi, B.J. & Heravi, M.J. (2007). SEAM+ Evolutionary Optimization Algorithm, *Proceedings of the 7th International Conference on Intelligent Systems Design and Applications*, IEEE Computational Intelligence Society , Rio de Janiro, Brazil.

Hallberg, J. & Peng, Z. (1996). Multicycle Scheduling under Local Timing Constraints using Genetic Algorithms and Tabu Search. Proceedings of 22nd Euromicro Conference'96, Beyond 2000: Hardware and Software Strategies, Prague, Czech Republic, 2-5 Sep. 1996.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor: University of Michigan Press.

Harik, G.R. (1997). *Learning Gene Linkage to Efficiently Solve Problems of Bounded Difficulty Using Genetic Algorithm*, PhD Dissertation, University of Illinois at Urbana-Champaign, Urbana, Illinois.

Harik, G.R.; Lobo, F.G. & Goldberg, D.E. (1998). The compact genetic algorithm. *Proceedings of the IEEE Conference on Evolutionary Computation*, 523-528.

Hou, T.; Wang, J.; Chen, L. & Xu, X. (1999). Automatic docking of peptides and proteins by using a genetic algorithm combined with a tabu search. *Protein Engineering,* Vol. 12, No.8, 639-647.

Jaeggi, D.M.; Parks, G.T.; Kipouros; T. & Clarkson, P.J. (2008) The development of a multiobjective Tabu Search algorithm for continuous optimization problems. European *Journal of Operational Research,* Vol. 185, No. 3, 16 March 2008, 1192-1212.

Kargupta, H. (1995). *SEARCH Polynomial Complexity And The Fast Messy Genetic*, PhD Dissertation, University of Illinois at Urbana-Champaign, Urbana, IL.

Larrañaga, P. & Lozano, J.A. (2002). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, Kluwer Academic Publishers.

Liang L.Y. & Chao, W.C. (2008) The strategies of tabu search technique for facility layout optimization, *Automation in Construction,* in press, available via Science Direct, doi:10.1016/j.autcon.2008.01.001.

Liu, Y.; Yia, Z.; Wu, H.; Ye, M. & Chen, K. (2008). A tabu search approach for the minimum sum-of-squares clustering problem. *Information Sciences,* in press, available via Science Direct, doi:10.1016/j.ins.2008.01.022.

Maynard Smith, J. & Szathmary, E. (1995) *The Major Transitions in Evolution*, WH Freeman: Oxford UK, 1995.

Merezhkovsky, K.S. (1909). The Theory of Two Plasms as the Basis of Symbiogenesis, a New Study or the Origins of Organisms. *Proceedings of the Studies of the Imperial Kazan University*, Publishing Office of the Imperial University, (In Russian).

Mermri, E.B; Katagiri, H.; Sakawa, M. & Kato, K. (2007). Remarks on the application of genetic algorithm and tabu search method to nonlinear spanning tree problems. *Applied Mathematics and Computation,* Vol. 188, No. 2, 15 May 2007, 1071-1086.

Mitchell, M. (1999). *An Introduction to Genetic Algorithms,* MIT Press, 0−262−13316−4, London, England.

Mühlenbein, H. & Mahnig, T. (1999). Convergence theory and application of the factorized distribution algorithm. *Journal of Computing and Information Technology*, Vol. 7, No. 1, 19-32.

Newman, D.R. (2006). *The Use of Linkage Learning in Genetic Algorithms*, http://www.ecs.soton.ac.uk/~drn05r/ug/irp/, (Last Seen: September 2006).

Oduntan, I.O.; Toulouse, M.; Baumgartner, R.; Bowman, C.; Somorjai, R. & Craini, T.G (2007), A multilevel tabu search algorithm for the feature selection problem in biomedical data. *Computers & Mathematics with Applications,* Vol. 55, No. 2, March 2008, 1019-1033.

Öncan, T.; Cordeau, J.F. & Laporte, G. (2007) A tabu search heuristic for the generalized minimum spanning tree problem. *European Journal of Operational Research,* in Press, available via Science Direct, doi:10.1016/j.ejor.2007.08.021.

Pacheco, J.; Alvarez, A.; Casado, S. & González-Velarde, J.L. (2007). A tabu search approach to an urban transport problem in northern Spain, *Computers & Operations Research,* in press, available via Science Direct, doi:10.1016/j.cor.2007.12.002.

Palubeckis, G. (2007). Iterated tabu search for the maximum diversity problem, *Applied Mathematics and Computation*, Vol. 189, No. 1, 1 June 2007, 371-383.

Pan, N.H.; Hsaio, P.W. & Chen, K.Y. (2007). A study of project scheduling optimization using Tabu Search algorithm, *Engineering Applications of Artificial Intelligence,* in Press, available via Science Direct, doi:10.1016/j.engappai.2007.11.006.

Pelikan, M.; Goldberg, D.E. & Cantu-Paz, E. (1999). BOA: The Bayesian optimization algorithm. Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, I, Orlando, FL. Morgan Kaufmann Publishers, San Francisco, CA., 525- 532.

Pelikan, M. & Goldberg, D.E. (2003). Hierarchical BOA solves using spin glasses and MAXSAT. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, II, 1275-1286.

Porto, S.C.S. & Ribeiro, C.C. (1996). Parallel tabu search message-passing synchronous strategies for task scheduling under precedence constraints. *Journal of Heuristics,* Vol. 1, No. 2, March 1996, 207-223.

Pothiya, S.; Ngamroo, I. & Kongprawechnon, W. (2007) Application of multiple tabu search algorithm to solve dynamic economic dispatch considering generator constraints. *Energy Conversion and Management,* Vol. 49, No. 4, April 2008, 506-516.

Potter, M.A., De Jong, K.A. (1994). A Cooperative Coevolutionary Approach to Function Optimization. *In: Parallel Problem Solving from Nature (PPSN III)*, Y. Davidor, H.-P. Schwefel and R. Manner (Eds.). Berlin: Springer-Verlag, 249-257.

Russle, S.J. & Norvig, P. (2002). *Artificial Intelligence: A Modern Approach, 2nd Edition*, Prentice-Hall, 111-112.

Sastry, K. & Goldberg, D. (2000). *On Extended Compact Genetic Algorithm* (IlliGAL Report No. 2000026). Urbana, IL: University of Illinois at Urbana-Champaign.

Shen, Q.; Shi, W.M. & Kong, W. (2007) Hybrid particle swarm optimization and tabu search approach for selecting genes for tumor classification using gene expression data, *Computational Biology and Chemistry,* Vol. 32, No. 1, February 2008, 53-60.

Vilcot, G. & Billaut, J.C. (2007). A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem, *European Journal of Operational Research,* in Press, available via Science Direct, doi:10.1016/j.ejor.2007.06.039.

Watcharasitthiwat, K.; Koseeyaporn, J. & Wardkein, P., (2006). Designing Digital FIR Filters Using Multiple Tabu Search Algorithm, *Proceedings of International Conference on Communications, Circuits and Systems,* pp. 171-175. ISBN: 0-7803-9585-9, Guilin, June 2006.

Watson, R.A. & Pollack, J.B. (1999), Incremental Commitment in Genetic Algorithms, *Proceedings of GECCO'99.*, Morgan Kaufmann, 710-717.

Watson, R.A. & Pollack, J.B. (2000). Symbiotic Combination as an Alternative to Sexual Recombination in Genetic Algorithms, *Proceedings of Parallel Problem Solving from Nature (PPSN VI)*, 425-436.

Zhu, Z.C.; Ng, K.M. & Ong, H.L. (2007) An Application of Tabu Search Algorithm on Costbased Job Shop Problem with Multiple Objectives, *Proceedings of IEEE International Conference on Industrial Engineering and Engineering Management,* pp. 912-916, ISBN: 978-1-4244-1529-8, Singapore, 2-4 Dec. 2007.

5

Tabu Search and Hybrid Genetic Algorithms for Quadratic Assignment Problems

Zvi Drezner
California State University, Fullerton
U.S.A.

1. Introduction

In this chapter experience with solving quadratic assignment problems is reported. The results reported in this chapter are the best results for heuristic solutions of the quadratic assignment problem available to date and can serve as bench mark results for future researchers who propose new approaches for solving quadratic assignment problems.
The most effective method to date for solving quadratic assignment problems heuristically is the hybrid genetic algorithm. The offspring produced by the genetic algorithms are improved by tabu search before considering them for inclusion into the population. Six different tabu searches are described and are embedded in a special genetic algorithm whose merging process is the most effective for heuristically solving quadratic assignment problems.
The most successful merging process (the crossover operator) used in the genetic algorithm is described. This specific merging process exploits the special structure of quadratic assignment problems and is especially effective when the distance matrix consists of "real" distances rather than random values.
A short cut suggested by Taillard (1995) is described. This short cut reduces the time required for the evaluation of all $O(n^2)$ values of the objective function by all pair-wise exchanges of facilities from $O(n^4)$ to $O(n^2)$ (i.e. $O(1)$ per pair exchange) where n is the number of facilities.
Grey pattern problems are quadratic assignment problems with a special structure. For these problems a special merging process and a special tabu search are developed (Drezner, 2006).
Several improvement schemes for genetic algorithms (or hybrid genetic algorithms) are described and discussed. These include: gender specific genetic algorithms, distance based approach to selecting parents in genetic algorithms, a distance based rule for removing population members, and compounded genetic algorithms. These improvement schemes can help researchers who work on other problems as well to improve the performance of their genetic or hybrid genetic algorithms.
The chapter concludes with summary tables of computational experiments with various techniques. These include the best known results for 32 "pure" quadratic assignment problems and 127 grey pattern quadratic assignment problems. All pure quadratic assignment problems have between 36 and 150 facilities. Smaller problems, with a few

exceptions, were optimally solved and thus not reported. One hundred and twenty six grey pattern problems have 256 facilities. One grey pattern problem with 64 facilities and 6 grey pattern problems with 256 facilities are optimally solved.

2. The quadratic assignment problem

The quadratic assignment problem (QAP) is considered one of the most difficult optimization problems to solve optimally. The QAP is a combinatorial optimization problem proposed by Koopmans & Beckmann (1957).
The problem is defined as follows. A set of n possible sites are given and n facilities are to be located on these sites, one facility at a site. Let c_{ij} be the cost per unit distance between facilities i and j and d_{ij} be the distance between sites i and j. The cost f to be minimized over all possible permutations, calculated for an assignment of facility i to site $p(i)$ for i=1, ... ,n, is:

$$f = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} d_{p(i)p(j)} \tag{1}$$

Optimal algorithms can solve relatively small problems (n≤36). Nug30, Kra30b, Tho30 were solved by Anstreicher et al. (2002); Kra30a by Hahn & Krarup (2001); Ste36a by Brixius and Anstreicher (2001); Ste36b, Ste36c by Nystrom (1999). Consequently, considerable effort has been devoted to constructing heuristic algorithms for its solution. The first heuristic algorithm proposed for this problem was CRAFT (Armour & Buffa, 1962) which is a descent heuristic. More recent algorithms use metaheuristics such as tabu search (Batiti & Tecchiolli, 1994; Skorin-Kapov, 1990; Taillard, 1991), simulated annealing (Burkard & Rendl, 1984; Wilhelm & Ward, 1987; Connoly, 1990; Misevicius, 2003), simulated jumping (Amin, 1999), genetic algorithms (Ahuja et al., 2000; Fleurent & Ferland, 1994; Tate & Smith, 1995; Drezner, 2003, 2005c), ant colony search (Gambardella et al., 1999), scatter search (Cung et al., 1997), or specially designed heuristics (Drezner, 2002; Li et al., 1994).
For a complete discussion and list of references see Burkard (1990), Cela (1998), Rendl (2002), Taillard (1995), and Drezner et al. (2005).

2.1 Grey pattern problems

Grey Pattern problems (Taillard, 1995) are a special class of quadratic assignment problems. These quadratic assignment problems have a special structure that can be exploited in the design of hybrid genetic algorithms.
The grey pattern problem (Taillard 1995) is based on a rectangle of dimensions n_1 by n_2. A grey pattern of m black points is selected from the $n = n_1 \times n_2$ slots in the rectangle while the rest of the slots remain white. This forms a "grey pattern" of density m/n. The objective is to have a grey pattern where the black points are distributed as uniformly as possible. This objective is achieved by defining a distance between pairs of points according to some rule. For more details see Taillard (1995).
Two grey pattern problems are available at QAPLIB http://www.seas.upenn.edu/qaplib. These are called Tai64c and Tai256c. Tai64c is a grey pattern problem in a square of 8 by 8 slots (n=64) and m=13 black points. Tai256c is a grey pattern problem in a square of dimensions 16 by 16 (n=256) and m=92 black points. Taillard and Gambardella (1997) define 126 grey pattern problems with the same distance matrix as Tai256c for n=256 selecting 3≤m≤128 black points.

The grey pattern quadratic assignment problems can be formulated in a simpler way (Taillard, 1995; Drezner, 2006). m slots out of n available slots need to be selected. d_{ij} is the distance between slots i and j. Let M of cardinality m be the subset of selected slots. The objective function, to be minimized by selection of the best subset M, is $f(M) = \sum_{i,j \in M} d_{ij}$. In Drezner (2006) the grey pattern problem is described as a special case of a more general problem. Consider n objects such as points in the plane or nodes of a network with a given distance between every pair of points. We wish to find a cluster of m points which minimizes the total distance between all pairs of points in the cluster. This cluster can be interpreted as the "tightest" cluster of m points.

Since this quadratic assignment formulation has a special structure, it is easier to solve as pointed out by Taillard (1995). Taillard (1995), Misevicius (2003a,b, 2004, 2005), and Drezner (2006) used this special formulation rather than the general QAP formulation for its solution.

3. Tabu searches for quadratic assignment problems

Tabu search procedures were suggested by Glover (1986). For a review of tabu searches the reader is referred to Glover & Laguna (1997). The search starts as a steepest descent algorithm but continues after the steepest descent algorithm has been terminated. Unlike the steepest descent, tabu search may take upward moves in the hope that a sequence of upward moves will lead to subsequent downward moves and eventually lead to a better solution. The direction of the search is determined by the recent history of moves that are "memorized". Once a move is performed, the reverse move (i.e. moving back to the previous combination) is forbidden for some iterations called tabu tenure (hence the name tabu which can also be spelled as taboo), thus pushing the search away from previous combinations. Imagine a search on a plane with many craters. One of these craters is the deepest one, and that one is the desired solution (the global optimum). The steepest descent performs only downward moves and may land at a shallow crater (a local optimum) and not at the global one. Tabu search attempts to get out of a shallow crater in the hope of getting to a better one. Therefore, when the steepest descent algorithm terminates at a bottom of a crater, upward moves are taken in tabu search while sliding back into the same crater is disallowed with the hope of sliding into deeper craters and eventually reaching the global optimum.

In this section six different tabu searches are presented: the robust tabu (RT) proposed by Taillard (1991), the modified robust tabu (MRT) suggested by Drezner & Marcoulides (2008), the simple tabu (ST) suggested in Drezner (2003) and improved by Drezner (2008a), the concentric tabu (CT) proposed by Drezner (2002), the ring moves (RM) and all moves (AM) suggested by Drezner (2005c).

3.1 Robust Tabu (RT)

Robust tabu (RT) was introduced by Taillard (1991) and is also described in Taillard (1995). The defined tabu list commonly used is set to contain pairs of facility-site (i.e., there are n^2 possible entries in the tabu list). There is a short term and long term tabu memory.

Short Term Memory: When a facility is removed from a site, the iteration number is recorded (meaning that the facility was at that site one iteration earlier). An exchange

between two facilities is disallowed (unless the objective function is better than the best one found so far) if at least one of the facilities moves back to a site it was removed from in the last u iterations. The tabu tenure u is randomly generated in $[0.9n, 1.1n]$ every iteration.
Long Term Memory: Every iteration after t iterations (we use $t=3n^2$): if there is an exchange between two facilities such that each facility moves to a site it was never there in the last t iterations, such an exchange preempts any other exchange and is executed. The long term memory serves as a diversification of the tabu search.

3.2 Modified Robust Tabu (MRT)

The modification suggested by Drezner & Marcoulides (2008) is replacing the range of $[0.9n, 1.1n]$ for the tabu tenure in the short term memory with the expanded range of $[0.2n, 1.8n]$. This modification yielded statistically proven superior results in computational experiments.

3.3 Simple Tabu (ST)

The simple tabu was suggested in Drezner (2003) and was modified to a wide range for generating the tabu tenure in Drezner (2008a).

1. The terminal solution of the descent heuristic is defined as the current solution and the best-known solution. The number of iterations of the descent heuristic is h. Empty the tabu list.
2. The following is repeated $\max\{2h, 50\}$ times:

- All pair-wise exchanges of facilities in the current solution are checked.
- If a solution better than the best-known solution is found, the best improving exchange is performed, the tabu list is emptied, and the next iteration starts.
- The tabu tenure TT is randomly generated in $[0.05n, 0.45n]$ and the tabu list consists of the facilities added to it in the last TT iterations.
- If no exchanged solution is better than the best-known solution, the best exchange (whether improving or not) between two facilities, *both* not in the tabu list, is performed.
- The two exchanged facilities are added to the tabu list.

3.4 Concentric Tabu (CT)

Concentric tabu search was introduced in Drezner (2002). One iteration of the concentric tabu search is very similar to the variable neighborhood search (Mladenovic & Hansen, 1997; Hansen & Mladenovic, 2001). The search is performed in "rings" around the center solution, proceeding from one ring to a larger one, and so on, until a pre-specified radius is obtained. A starting solution is selected as the center solution. Every feasible solution of the quadratic assignment problem is a permutation p of the center solution. A "distance" Δp is defined for each solution p (permutation of the center solution). The distance Δp is the number of facilities in p that are not in their center solution site. Note that $\Delta p \le n$. The tabu list consists of solutions that are not farther than Δp from the center solution, thus forcing the search away from the center solution.
For each Δp three solutions are recorded: s_0, s_1, and s_2. The solution s_0 is the best encountered solution with distance Δp. Similarly, s_1 and s_2 are the best encountered solutions with distances $\Delta p+1$, and $\Delta p+2$, respectively. The depth of the search is set to $d \le n$.

One Iteration of CT

1. Set Δp=0. The starting solution s_0 is the center solution and the best found solution.
2. All pair exchanges of s_0 are evaluated.
3. If the exchanged solution is better than the best found solution, the best found solution is updated and the rest of the exchanges are evaluated.
4. If the distance of an exchanged solution is Δp or lower, it is in the tabu list. It is ignored and the rest of the exchanges are evaluated.
5. If its distance is Δp+1 or Δp+2, s_1 or s_2 are updated if necessary.
6. If a new best found solution is found by scanning all the exchanges of s_0, the starting (center) solution is set to the new best found solution. Go to Step 1.
7. Otherwise, set s_0=s_1, s_1=s_2, and s_2 is emptied. Set Δp=Δp+1.
8. If Δp=d+1 stop the iteration. Otherwise, go to Step 2.

3.5 Two extended concentric tabu searches

Two variants of the concentric tabu search are suggested in Drezner (2005c): ring moves (RM) and all moves (AM). These extended concentric tabu searches are based on the following observations.

Consider all possible changes in Δp ($\Delta\Delta p$) when facilities f1 and f2 are exchanged. The nine possible changes are depicted in Table 1. "Back" means that the facility is moved back to its center site (decreasing Δp by 1); "Out" means that a facility is removed from its center site (increasing Δp by 1), and NC (No Change) means that the facility was neither in its center site nor moved into its center site so Δp is not changed. The combination of one "Back" and one "Out" is impossible. If facility f1 is moved out of its center site, facility f2 could not be moved into its center site because this site is the center site of facility f1.

f1	f2	$\Delta\Delta p$
Back	Back	-2
Out	Back	*
NC	Back	-1
Back	Out	*
Out	Out	+2
NC	Out	+1
Back	NC	-1
Out	NC	+1
NC	NC	0

* Impossible

Table 1. The nine possible moves

In Table 2 we summarize the possibilities in a different way.

$\Delta\Delta p$	Move
-2	Both Back
-1	One Back, the other NC
0	Both NC
+1	One out, the other NC
+2	Both Out

Table 2. Summary of moves

In the original concentric tabu (CT) only +1 and +2 moves are considered (unless a solution better than the center solution is found). Moves of -2, -1, and 0 are in the tabu list. This means that a facility that was taken out from its center location, cannot be moved back into its center location throughout the iteration because "Back" moves are not considered (except, of course, when a better best known solution is found). Also, every move must include a facility taken out of its center location. These restrictions are a bit "harsh". Two possible modifications to the concentric tabu are suggested. These modifications reduce the tabu list and yet guarantee that no cycling is possible.

3.6 Ring Moves (RM)

In the RM algorithm, we consider also moves "inside the ring". A change $\Delta\Delta p = 0$ means that the exchanged facilities did not move in or out of their center location. The solution following the exchange has the same Δp (is in the same ring).

One Iteration of RM

1. Set $\Delta p=0$. The starting solution s_0 is the center solution and the best found solution.
2. All pair exchanges of s_0 are evaluated.
3. If the exchanged solution is better than the best found solution, the best found solution is updated and the rest of the exchanges are evaluated.
4. If the distance of an exchanged solution is lower than Δp, it is in the tabu list. It is ignored and the rest of the exchanges are evaluated.
5. If its distance is Δp, $\Delta p+1$ or $\Delta p+2$, s_0, s_1 or s_2 are updated if necessary. Note that the original s_0 is still used for the rest of the pair exchanges.
6. If a new best found solution is found by scanning all the exchanges of s_0, the starting (center) solution is set to the new best found solution. Go to Step 1.
7. Otherwise,
   - If s_0 has changed, go to Step 2.
   - If s_0 has not changed, set $s_0=s_1$, $s_1=s_2$ and s_2 is emptied. Set $\Delta p=\Delta p+1$.
   - If $\Delta p=d+1$ stop the iteration. Otherwise, go to Step 2.

The algorithm allows for exchanges between two facilities, both with "No Change" that improve the present value of the objective function. Note that when a facility was taken out from its center location, it cannot be moved back into its center location throughout the iteration. Therefore, the ring moves do not rectify this issue encountered in concentric tabu.

3.7 All Moves (AM)

In the AM algorithm the tabu list is eliminated and replaced by a different approach. A list of the best encountered solution for each $0\leq\Delta p\leq$n is maintained (at the beginning only the center solution is in the list). Members in the list are tested to find whether their neighbors are better than other members in the list or themselves. A list member whose neighbors were not tested yet, is flagged. Once none of the members in the list are flagged, the iteration terminates. This way the value of Δp can change up and down while in CT it can only increase and in RM it can increase or stay the same.

One Iteration of AM

1. Set $\Delta p=0$. The best encountered solution for each $2\leq\Delta p\leq n$ is emptied and all flags removed. The starting solution is the best encountered for $\Delta p=0$ and is flagged.
2. The flagged solution, if there is one, with the lowest Δp is selected for pair exchanges.

3. If there are no flagged solutions left, the iteration terminates with the center solution.
4. All pair exchanges of the selected solution are evaluated and its flag removed.
5. If the exchanged solution is better than the best found solution, the best found solution is updated and the rest of the exchanges are evaluated.
6. If an exchanged solution is better than the best encountered solution for the appropriate Δp, it replaces it and flagged. (If the improvement is for the same Δp, the original selected solution is kept for the remaining pair exchanges).
7. If a new best found solution is found by scanning all the exchanges of the selected solution, the starting (center) solution is set to the new best found solution. Go to Step 1.
8. Otherwise, go to Step 2.

This variant has no tabu list. It allows moves to a ring with a smaller Δp, if they improve the best encountered value of the objective function for that Δp. This variant may allow a facility that was removed from its center location to move back to it.

3.8 The improvement procedure

Robust Tabu (RT), the modified robust tabu (MRT) and the simple tabu (ST) constitute the improvement procedure. One iteration of concentric tabu (CT), ring moves (RM), or all moves (AM) is repeated L times called "levels" in the post merging improvement procedure of the hybrid genetic algorithm. In Drezner (2003) concentric tabu with L=3 levels (CT3) was used.

The Improvement Procedure for CT, RM, and AM

1. The result of the merging procedure is the center solution. It is also the best found solution.
2. Set a counter c=0.
3. Select d randomly in [n-4, n-2] (other ranges for d may be used). Perform one iteration of CT, RM, or AM on the center solution.
4. If the iteration improved the best found solution go to Step 2.
5. Otherwise, advance the counter c=c+1, and
   - If $c \le L$ and is odd use the best solution with depth d as the new center solution and go to Step 3.
   - If $c \le L$ and is even use the best solution found throughout the scan (the previous center solution is not considered) as the new center solution and go to Step 3.
   - If c=L+1 stop and report the best found solution.

In order to reduce run time, a shorter depth d randomly generated in [0.3n, 0.9n] defined as a "short" search was suggested in Drezner (2005a). It produced improved results when the number of levels was relatively small but was inferior for larger number of levels reported in this chapter.

3.9 Selecting among equal values

In many heuristic optimization algorithms (such as tabu search), each iteration the best "move" to another solution to be employed in the next iteration is selected. It is very common that there are several moves with exactly the same value for the objective function. Which of the tying moves should be selected? If we select a move only if it is better than the best move found so far, the first tying move will always be selected. If we select a move as long as it is not worse than the best move found so far, the last one will be selected. This

may bias the search (for example, if the moves are scanned by the order of the facilities) giving a preference to either early scanned moves or late scanned moves. One possible approach is to evaluate the possible moves in random order. Another way is to save the tying moves and once the process is completed, when we know how many tying moves there are, a tying move is selected at random. Both of these approaches are cumbersome and require extra code and memory space. The following approach (Drezner, 2008b) is a simple approach which is very easy to implement.

When the moves are evaluated et seriatim, we do not know how many tying moves there will eventually be. The strategy is to select tying move number K with a probability of $1/K$. The first move which is better than the best move found so far is selected with probability 1. The second tying move is selected with probability of 1/2, the third with probability of 1/3 and so on. This rule is obvious for one or two tying moves (if there are 2 tying moves, each is selected with a probability of 50%). It is proven in Drezner (2008b) by mathematical induction that if eventually there are K tying moves, each of them is selected with a probability of $1/K$.

3.10 A short cut for reducing the run time in tabu searches

Taillard (1995) suggested an effective short cut for reducing the run time necessary for evaluating the values of the objective function for all pair exchanges. There are $n(n\text{-}1)/2$ possible pair exchanges and evaluating each value of the objective function requires $O(n^2)$ time leading to a total of $O(n^4)$ time. Taillard (1995) suggested the following procedure that calculates *all* these values of the objective functions in $O(n^2)$ time. Run times of the various tabu searches and the hybrid genetic algorithms were reduced by a substantial factor using this technique. This short cut can be applied for the calculations of pair exchanges in all six tabu searches described above.

Since we experimented only with symmetric problems, we present this short cut for symmetric problems with zero diagonal (i.e., the cost between a facility and itself, and the distance between the same two locations is zero). It can be easily generalized to non symmetric problems.

Let Δf_{rs} be the change in the cost f, calculated by Equation (1), by exchanging the sites of facilities r and s. This is a concept similar to the derivative of f. There are $n(n-1)/2$ such values. It can be easily verified by examining Equation (1) that:

$$\begin{aligned}\Delta f_{rs} &= 2\sum_{i=1}^{n}\left\{c_{ir}\left[d_{p(i)p(s)}-d_{p(i)p(r)}\right]+c_{is}\left[d_{p(i)p(r)}-d_{p(i)p(s)}\right]\right\}= \\ &= 2\sum_{i=1}^{n}\left\{\left[c_{ir}-c_{is}\right]\left[d_{p(i)p(r)}-d_{p(i)p(s)}\right]\right\}\end{aligned} \tag{2}$$

Calculating Δf_{rs} by using Equation (2) requires only $O(n)$ time rather than $O(n^2)$ time required to calculate f by Equation (1). Taillard (1995) points to yet a faster formula for calculating Δf_{rs}.

Let $\Delta_{uv}f_{rs}$ be the change in the value of the objective function between the exchanged permutation by uv, and an additional exchanged pair rs when rs and uv are mutually exclusive. Note that $\Delta_{uv}f_{rs}=\Delta_{rs}f_{uv}$. This is a concept similar to the second derivative of f. This change in the value of the objective function can be calculated in O(1) time (starting from the second iteration) if the pairs rs and uv are mutually exclusive. The formula is based on Δf_{uv}

(the change in the value of the objective function from the previous permutation by exchanging the pair uv). Therefore, one needs to keep all the values of Δf_{ij} for all i, j. Saving these Δf_{ij} values requires $O(n^2)$ time for each evaluation of all pair exchanges of s_0.
Since

$$\Delta f_{uv} = 2\sum_{i=1}^{n}\left\{\left[c_{iu} - c_{iv}\right]\left[d_{p(i)p(u)} - d_{p(i)p(v)}\right]\right\}$$

it can be easily verified that:

$$\Delta_{uv} f_{rs} = \Delta f_{uv} + 2\left[c_{su} + c_{rv} - c_{sv} - c_{ru}\right]\left[d_{p(s)p(u)} + d_{p(r)p(v)} - d_{p(s)p(v)} - d_{p(r)p(u)}\right] \quad (3)$$

which is calculated in O(1). Note that only $2n - 3$ pairs are not mutually exclusive and formula (2) can be used in these cases to evaluate $\Delta_{uv} f_{rs}$. Therefore, evaluating the change in the value of the objective function for all $n(n-1)/2$ possible pair exchanges (which is required for one step of each of the tabu search algorithms described above) requires $O(n^2)$ time rather than $O(n^4)$ time by calculating each f directly or $O(n^3)$ time by using only reduction scheme (2).

4. Hybrid genetic algorithms

Genetic algorithms (Holland, 1975; Goldberg, 1989) simulate evolution and survival of the fittest. A population (made of individual permutations) evolves over time (generations). Pairs of population members (permutations) mate and produce an offspring (two permutations are merged to produce a new permutation). Good offspring are kept in the population whereas unfit population members are discarded (the survival of the fittest). The population evolves and at the end of the process, the population usually consists of fairly good solutions (without a guarantee that the optimal solution is found). Hybrid genetic algorithms, sometimes called memetic algorithms (Moscato, 2002), incorporate some improvement heuristic on every offspring before considering its inclusion into the population. For a review see Drezner & Drezner (2005).
The following is a short description of the specific hybrid genetic algorithm used for the computational experiments.

1. A starting population of size P is randomly selected, and the improvement procedure is applied on each starting population member.
2. Steps 3-6 are repeated for a pre-specified number of G generations.
3. Two population members are randomly selected and merged by a crossover operator to produce an offspring.
4. The improvement procedure is applied on the merged solution, possibly improving it.
5. If the value of the objective function of the offspring is not better than the value of the objective function of the worst population member, the offspring is ignored and the process of the next generation starts.
6. Otherwise,
   - If the offspring is identical to an existing population member, it is ignored and the process of the next generation starts.

- If the offspring is different from all population members, the offspring replaces a population member selected according to one of the rules described in Section 5.4. In most variants the worst population member is removed.

There are many genetic algorithms suggested for the solution of the quadratic assignment problem (Ahuja et al., 2000; Fleurent & Ferland, 1994; Tate & Smith, 1995; Drezner, 2003, 2005c). Drezner (2003) suggested a merging process that proved to be most effective for the solution of quadratic assignment problems. There are two merging processes suggested in Drezner (2003): the cohesive merging process and the scrambled merging process. The cohesive merging process was slightly better than the scrambled merging process and was used in subsequent algorithms. We present here the cohesive merging procedure.

4.1 The cohesive merging procedure

The most crucial part of a successful genetic algorithm is the merging process of two parents to produce an offspring. For the process to be effective an offspring should be significantly better (in terms of its value of the objective function) than a randomly generated solution. Otherwise, we do not gain by the merging process. It is true that such an algorithm may find a good solution, but it does not have a significant advantage over repeating the improvement process from randomly generated solutions the same number of times. Therefore, it is essential to find a merging rule that exploits the structure of the problem and is likely to use "good features" of the parents when creating an offspring.

The cohesive merging process for the solution of the quadratic assignment problem is similar to the successful merging procedure used in Drezner & Salhi (2002) for solving a network design problem. Drezner & Wesolowsky (1997) suggested the problem of designing a network so that each link can be either a two way link or a one-way link in one direction. The model was extended in Drezner & Wesolowsky (2003) to include the option of eliminating links. The problem is to determine the design of each link to minimize an objective function. A hybrid genetic algorithm is proposed in Drezner & Salhi (2002) for the solution of the problem. Suppose that two parents are selected and an offspring is designed. The network is divided into two cohesive parts by selecting a pivot node, assigning a count of 1 to all links directly connected to it, a count of 2 to all links connected to a link of count 1, and so on. Each link gets a count. The median of these counts for all links is calculated. The design of links with a count below the median is taken from the first parent and the design of links above the median is taken from the second parent. The design of links that have a count equal to the median is randomly selected from one of the parents. This way each part of a parent is a connected part of the network. Drezner & Salhi (2002) suggest considering the n possible partitions (one for each pivot node) and selecting the best offspring of these n partitions for an improvement procedure and possible inclusion in the population. For the quadratic assignment problem a similar merging process is suggested (Drezner, 2003).

In the cohesive merging process we attempt to divide the sites into two cohesive parts where each has all its facilities from the same parent. A pair of parents is randomly selected. The parent with the better value of the objective function is selected as the first parent. If the two parents tie in the value of the objective function, one of them is arbitrarily selected as the first parent. The following is executed for every pivot site.

1. The median distance from the pivot site to all sites is calculated (this is done in the preamble and not at every iteration).

2. A site that is closer than the median to the pivot site is assigned the facility from the first parent.
3. All other sites are assigned the facility from the second parent.
4. It is likely that some facilities are assigned twice and some are not assigned at all. Therefore,
   - Create a list of unassigned facilities.
   - Find all facilities that are assigned twice, and replace the site that is farther than the median (i.e., from the second parent) with a facility from the list.
   - Remove the selected facility from the list.
5. This completes the merge of the two selected parents for one pivot site.

The merged solution with the lowest value of the objective function is the offspring selected for the improvement algorithm.

4.2 Merging processes for the grey pattern problems

Grey pattern problems (Taillard, 1995; Drezner, 2006) are special cases of the quadratic assignment problem and special merging processes designed for it are warranted. We present the descent merging process and its extension to a tabu merging process (Drezner, 2006). These merging processes do not resemble neither the standard crossover operator nor the hybrid genetic algorithm approach. They combine elements of both. The tabu merging process provided the best results.

4.3 The descent merging process for grey pattern problems

The descent merging process is similar to the merging process suggested in Berman and Drezner (2007).

1. The two parents are M_1 and M_2, each represented by a set of m slots.
2. The intersection between M_1 and M_2 is: $M^I = M_1 \cap M_2$. The cardinality of the intersection is m^I.
3. The union of M_1 and M_2 is: $M^U = M_1 \cup M_2$. The cardinality of M^U is $2m - m^I$.
4. K different slots not in M^U are randomly selected to form M^K (if $2m - m^I + K > n$, only $n - 2m + m^I$ points are selected).
5. All the points in M^U which are not in M^I define M^E. The cardinality of M^E is $2m - 2m^I$.
6. Define $M^D = M^E \cup M^K$. The cardinality of M^D is $m^D = \min\{n - m^I, 2m - 2m^I + K\}$.
7. A starting offspring M' of cardinality m is created by randomly adding to M^I $m - m^I$ points from M^D.
8. A restricted descent process is performed on M' by adding or removing only points in M^D and keeping the points in M^I in the selected set.
9. The result of the restricted descent process is the offspring.

4.4 The tabu merging process

We also experimented with a tabu extension of the restricted descent search. Let h be the number of iterations performed by the restricted descent algorithm. A restricted tabu search for additional $5h$ iterations is performed. The value K=3 was used in the descent algorithm and the tabu search. The tabu tenure was randomly generated in the range $[0.02(n\text{-}m), 0.2(n\text{-}m)]$. We need to select $m - m^I$ slots out of m^D slots. If $m^D - (m - m^I) \leq 5$, the tabu search is not performed and the result of the descent algorithm is applied. For complete details see Drezner (2006).

5. Improvements of genetic algorithms

The improvements described below are described in Drezner & Drezner (2005). The compounded genetic algorithm is proposed in Drezner (2005a), the gender specific genetic algorithm is proposed in Drezner & Drezner (2006), the distance based parent selection is proposed in Drezner & Marcoulides (2003), and the modification of the removal rule of population members is proposed in Drezner (2005b). The reader is referred to these papers for a complete description of the improvements. There are many other improvements suggested by many authors. For example, mutations (Spears, 2000), invasions (Goldberg, 1989), parallel genetic algorithms (Cantu-Paz, 1998), among others.

5.1 Compounded genetic algorithms

In the compounded genetic algorithm (Drezner, 2005a) genetic algorithms are applied in two phases, generating the starting population for phase 2 by repeating genetic algorithms in phase 1. This mimics evolving parallel populations at several isolated locations. The best species in each location are moved to a common location thus creating a "high quality" starting population. Suppose that a population of P members is applied in phase 2. Genetic algorithms are run K times in phase 1 using K randomly generated starting populations (it is convenient but not necessary to have integer P/K). The population size of the phase 1 genetic algorithm should have at least P/K members. The best P/K population members of each run are compiled to construct the starting population for phase 2. Phase 2 genetic algorithm is run once. It is recommended that a "quick" genetic algorithm is used for phase 1 and an "effective" and possibly "slow" genetic algorithm is used for phase 2.

For example, if a population of P=100 members is required for phase 2, phase 1 can be run K=20 times (each with a population of at least 5 members), the best P/K=5 population members are selected from each run and compiled to create a starting population for phase 2.

Note that the best solution found in phase 1 by any of the runs can only be improved by the compounded genetic algorithm because the best solution found during phase 1 is a member of the starting population of phase 2 and can only be removed from the population by better solutions.

5.2 Gender specific genetic algorithms

In nature, most advanced species require two genders in order to mate and reproduce. The gender modification attempts to mimic this natural process. One can argue that the division into two genders was selected over time as the preferred way for producing offspring and is therefore superior to other possible mating schemes. Epelman et al. (2005) show that having only two genders maximizes long run viability. Their finding is not directly related to genetic algorithms. However, it supports our experience that it is true for genetic algorithms as well. In gender-specific genetic algorithms the diversity of the population is better maintained with no detrimental effects on run time.

It is easy to "convert" a given genetic algorithm to a gender-specific one. Three minor modifications are required (Drezner & Drezner, 2006).

1. When the starting population is generated, half the population members are designated as males and half are designated as females. The assignment of gender is done at random and no characteristic of the population member is used for such determination.

2. When selecting two parents, the first parent is randomly selected while the second is randomly selected from the pool of the opposite gender.
3. When an offspring is generated, it is randomly assigned a gender with a 50% probability of being assigned a male gender and 50% probability a female. Again, no characteristic of the offspring should be used to determine its gender.

No extra effort is required for the implementation of the gender-specific modification. A vector of genders for population members needs to be maintained, along with the gender determined for each offspring. In Drezner & Drezner (2006) it has been statistically shown that the gender-specific algorithm significantly improves the solutions on four sets of optimization problems.

Note that it is important that an offspring's gender is randomly determined. An early attempt (Allenson, 1992) for such a modification failed because it was suggested that the offspring is assigned the gender of the discarded population member. The rationale for this rule is to keep the population half males and half females. However, such a rule is inconsistent with nature. The concern is that if the population becomes all males or all females no further evolution is possible. The evolutionary process must be terminated prematurely if such a population structure evolves. In Drezner & Drezner (2006) it is shown that for a sufficiently large population (50 or more members), the probability that all population members will have the same gender is extremely low and such an event can be ignored.

5.3 Distance based parent selection

All human cultures prohibit marriage between siblings or between parents and children (genetically similar pairs). In societies where marriages are arranged, similarity in socio-economic standing, but not genetic make-up, is prevalent. Some plants avoid pollination from genetically similar or identical individuals because self-pollination or pollination by 'siblings' is typically unsuccessful, a phenomenon referred to in biology as "inbreeding depression". Mating between close relatives often results in less fit offspring. Another, less well known biological fact, is that mating between genetically distant members of the same species can lead to a decline in offspring fitness, a condition known as "outbreeding depression" or "hybrid breakdown". Some species avoid pollination from individuals that are geographically distant or genetically dissimilar, as offspring may be less suited to the local conditions and may be poorer competitors locally. Edmands (1999, 2002) observed that parental divergence (parents who are genetically distant) leads to less fit offspring.

In genetic algorithms, if dissimilar individuals mate, the offspring is more genetically diverse which is critical in maintaining a population's genetic diversity. However, parents who are too dissimilar produce less fit offspring. Using the distance criterion for parent selection Drezner & Marcoulides (2003) crafted a rule attempting to find dissimilar but not too distant parents. A parameter K=1,2,3,…. is used. The first parent is randomly selected and K candidates for mating are then randomly selected. The distance (number of variables with different values, the Hamming distance metric) between the first parent and all candidate mates is calculated. The farthest mate among these K candidates is selected as the second parent. Note that K=1 is the "standard" parent selection. Drezner & Marcoulides (2003) found that the efficiency of the modification for a set of test problems peaks for K=2,

3. Selecting the farthest population member as a mate does not work well. Two dissimilar parents may produce an offspring which is not improved compared with a randomly generated solution. It was also found that run time increases with K which further reduces the appeal of larger values of K.

5.4 Removal of population members

In most standard genetic algorithms, when an offspring is generated, it is compared with the worst population member and if the offspring is better than the worst population member, it replaces it. Some genetic algorithms employ a rule according to which if the offspring is identical to an existing population member it is not considered for inclusion in the population. This precludes the possibility of having two identical population members. Drezner (2005b) suggested a different rule for removal of population members. Two rules for removal of a population member, once a better offspring (who is not identical to an existing population member) is found, are used. Rule 1 is the standard approach and Rule 2 is a new one.

Rule 1: Remove the worst population member.

Rule 2: Hamming distances between all pairs of population members are calculated. Suppose that the shortest distance among all pairs of population members is d. All existing population members who are at distance d from another population member form a subset. This subset must have at least two members (at least one pair of population members are at distance d from one another). Remove the worst population member in this subset.

In the experiments performed in Drezner (2005b) it was found that Rule 2 is not necessarily better than Rule 1. The suggested rule is to select Rule 2 with probability p, and to select Rule 1 otherwise. Note that p=0 is the standard rule (Rule 1), and p=1 is Rule 2. The mix between the two rules by selecting $0<p<1$, seems to work well.

6. Computational results

All the results reported in this chapter are based on programs coded in Fortran, compiled by Intel 9.0 Fortran compiler and ran on a 2.8GHz Pentium IV desktop computer with 256MB of RAM.

6.1 Results for the first set of problems

In Table 3 we report the results obtained in Drezner (2008a) for 18 problems available in the QAPLIB. These problems are: Ste36a, Ste36c (Steinberg, 1961), Tho40 (Thonemann & Bolte, 1994), Sko49 (Skorin-Kapov, 1990), Wil50 (Wilhelm & Ward, 1987), Sko56, Sko64, Sko72, Sko81, Sko90, Sko100a-f (Skorin-Kapov, 1990), Wil100 (Wilhelm & Ward, 1987), and Tho150 (Thonemann & Bolte, 1994).

Each problem was solved twenty times. The results for the first 17 problems are by the hybrid genetic algorithm using the modified robust tabu search with 60 levels (MRT60). The results for the last problem (Tho150) are by a simple tabu hybrid genetic algorithm (Drezner, 2008a) with 100 levels. When MRT60 was applied to Tho150, the best known solution was found 6 times out of 20 runs with the average solution of 0.002% over the best known solution requiring a run time of 1223.57 minutes.

#	Problem	n	Best Known	(1)	(2)	Time (min.)
1	Ste36a	36	9,526	20	0.000%	1.55
2	Ste36c	36	8,239.11	20	0.000%	1.55
3	Tho40	40	240,516	20	0.000%	2.12
4	Sko49	49	23,386	20	0.000%	4.27
5	Wil50	50	48,816	20	0.000%	4.55
6	Sko56	56	34,458	20	0.000%	7.15
7	Sko64	64	48,498	20	0.000%	12.41
8	Sko72	72	66,256	20	0.000%	19.85
9	Sko81	81	90,998	20	0.000%	31.94
10	Sko90	90	115,534	20	0.000%	48.46
11	Sko100a	100	152,002	20	0.000%	73.57
12	Sko100b	100	153,890	20	0.000%	73.47
13	Sko100c	100	147,862	20	0.000%	73.46
14	Sko100d	100	149,576	20	0.000%	73.50
15	Sko100e	100	149,150	20	0.000%	73.47
16	Sko100f	100	149,036	18	0.001%	73.48
17	Wil100	100	273,038	20	0.000%	73.57
18	Tho150	150	8,133,398	17	0.000%	1949.05

(1) Number of times (out of 20 runs) that best known solution found
(2) Percent of average solution above best known solution

Table 3. Results for first set of problems

6.2 Results for de Carvalho and Rahmann problems

Recently de Carvalho & Rahmann (2006) introduced a new class of quadratic assignment problems that turn out to be extremely difficult to solve. There are 14 problems in this set. Seven problems called border length minimization and seven problems called conflict index minimization. The costs of the seven conflict minimization are not symmetric but the distances are and the diagonal elements are zeroes. In order to solve these problems by our symmetric program, the costs are redefined as $c'_{ij} = c'_{ji} = c_{ij} + c_{ji}$. All these 14 problems were solved by GRASP (Oliveira Pardalos & Resende, 2004) reported in de Carvalho & Rahmann (2006), GATS (genetic algorithm and tabu search) solved by the method in Rodriguez et al., (2004), EDA (estimation of distribution algorithms) reported in Pelikan et al. (2007), and Drezner & Marcoulides (2008). All these researchers report that these problems are extremely difficult to solve among the benchmark problems available on QAPLIB. Drezner & Marcoulides (2008) obtained the best results by applying the modified robust tabu search.

There are two parameters that determine the total run time required by the hybrid genetic algorithm: the number of generations and the depth of the tabu search applied in each generation. The run time of the algorithm is proportional to each of these parameters and thus proportional to their product. The "standard" number of generations (Drezner, 2003) is max{20n, 1000} thus we used G×max{20n, 1000} generations. The depth of the tabu search (the number of iterations in the tabu search) is D×n. G and D are parameters. Various values of G and D such that G×D=120 were tested to determine the trade-off between them. In

Table 4 the best known results for this set of problems are reported along with average run times reported in Drezner and Marcoulides (2008). Twelve of these results are better than the best results in all previous reports.

n	Border Length Minimization		Conflict Index Minimization	
	Best Known	Time (min.)	Best Known	Time (min.)
36	3,296	1.89	168,611,971	1.89
49	4,548	4.92	236,355,034	4.92
64	5,988	13.92	325,671,035	13.91
81	7,536	37.98	427,447,820	38.14
100	9,272	95.28	523,146,366	95.53
121	11,412	235.61	653,416,978	235.29
144	13,472	524.04	795,009,899	525.57

Table 4. de Carvalho and Rahmann problems

6.3 Results for grey pattern problems

The best known values of the 126 grey pattern problems are reported in Table 5. The original best known values are reported in Taillard and Gambardella (1997). Misevicius (2003a,b, 2004, 2005) improved some best known values. Eight additional improved best known values are reported in Drezner (2006) by using the hybrid genetic algorithm with the tabu merging process. Drezner (2006) also found all previously best known values. Average run time was 3.25 minutes per problem. In Drezner (2006) it was proven that the solution of 1,855,928 for the Tai64c problem is optimal and the six solutions to Tai256c with m=3-8 reported in Table 5 are also optimal.

7. Conclusions

In this chapter we report the best known results for 159 quadratic assignment problems. Thirty two of these problems are "pure" quadratic assignment problems and 127 of them are grey pattern problems which are a specific type of quadratic assignment problems. These results were obtained by hybrid genetic algorithms using tabu search as its improvement procedure. The genetic algorithm and six variants of tabu search are described and implemented for obtaining these best known solutions. A short cut for calculating the change in the value of the objective function by exchanging pairs of facilities, and an effective merging procedure for genetic or hybrid genetic algorithms are described. Special hybrid genetic algorithms that exploit the special structure of grey pattern problems are designed for solving these problems.

We also describe four improvements to genetic or hybrid genetic algorithms, and an easy way to randomly select a solution among equally valued options. We also observe that increasing the range from which the tabu tenure is randomly selected is also beneficial. All these improvements can be used in designing tabu search, genetic or hybrid genetic algorithms, for heuristically solving any optimization problem.

m	BK	m	BK	m	BK	m	BK
3	7,810	35	4,890,132	67	21,439,396	99	52,660,116
4	15,620	36	5,222,296	68	22,234,020	100	53,838,088
5	38,072	37	5,565,236	69	23,049,732	101	55,014,262
6	63,508	38	5,909,202	70	23,852,796	102	56,202,826
7	97,178	39	6,262,248	71	24,693,608	103	57,417,112
8	131,240	40	6,613,472	72	25,529,984	104	58,625,240
9	183,744	41	7,002,794	73	26,375,828	105	59,854,744
10	242,266	42	7,390,586	74	27,235,240	106	61,084,902
11	304,722	43	7,794,422	75	28,114,952	107	62,324,634
12	368,952	44	8,217,264	76	29,000,908	108	63,582,416
13	457,504	45	8,674,910	77	29,894,452	109	64,851,966
14	547,522	46	9,129,192	78	30,797,954	110	66,120,434
15	644,036	47	9,575,736	79	31,702,182	111	67,392,724
16	742,480	48	10,016,256	80	32,593,088	112	68,666,416
17	878,888	49	10,518,838	81	33,544,628	113	69,984,758
18	1,012,990	50	11,017,342	82	34,492,592	114	71,304,194
19	1,157,992	51	11,516,840	83	35,443,938	115	72,630,764
20	1,305,744	52	12,018,388	84	36,395,172	116	73,962,220
21	1,466,210	53	12,558,226	85	37,378,800	117	75,307,424
22	1,637,794	54	13,096,646	86	38,376,438	118	76,657,014
23	1,820,052	55	13,661,614	87	39,389,054	119	78,015,914
24	2,010,846	56	14,229,492	88	40,416,536	120	79,375,832
25	2,215,714	57	14,793,682	89	41,512,742	121	80,756,852
26	2,426,298	58	15,363,628	90	42,597,626	122	82,138,768
27	2,645,436	59	15,981,086	91	43,676,474	123	83,528,554
28	2,871,704	60	16,575,644	92	44,759,294	124	84,920,540
29	3,122,510	61	17,194,812	93	45,870,244	125	86,327,812
30	3,373,854	62	17,822,806	94	46,975,856	126	87,736,646
31	3,646,344	63	18,435,790	95	48,081,112	127	89,150,166
32	3,899,744	64	19,050,432	96	49,182,368	128	90,565,248
33	4,230,950	65	19,848,790	97	50,344,050		
34	4,560,162	66	20,648,754	98	51,486,642		

Table 5: Best known values for grey pattern problems

8. References

Ahuja, R.K.; Orlin, J.B. & Tiwari, A. (2000). A Descent Genetic Algorithm for the Quadratic Assignment Problem. *Computers and Operations Research*, 27, 917-934.

Allenson R. (1992). Genetic algorithms with gender for multi-function optimisation. Technical Report EPCC-SS92-01, Edinburgh Parallel Computing Centre, Edinburgh, Scotland, 1992.

Amin S. (1999). Simulated Jumping. *Annals of Operations Research*, 86, 23-38.

Anstreicher, K.; Brixius, N.; Goux, J.-P & Linderoth, J. (2002). Solving large quadratic assignment problems on computational grids. *Mathematical Programming*, 91, 563-588.

Armour, G.C. & Buffa, E.S. (1963). A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities. *Management Science*, 9, 294-309.

Battiti, R. & Tecchiolli, G. (1994). The Reactive Tabu Search, *ORSA Journal on Computing*, 6, 126-140.

Berman O. & Z. Drezner (2007). The multiple server location problem. *Journal of the Operational Research Society*, 58, 91-99.

Brixius N.W. & K.M. Anstreicher (2001). The Steinberg wiring problem. Working Paper, The University of Iowa, October 2001.

Burkard, R.E. (1990). Locations with Spatial Interactions: The Quadratic Assignment Problem. In *Discrete Location Theory* P.B. Mirchandani & R.L. Francis (Eds.), 387-437, Wiley, Berlin.

Burkard, R.E. & Rendl, F. (1984). A Thermodynamically Motivated Simulation Procedure for Combinatorial Optimization Problems. *European Journal of Operational Research*, 17, 169-174.

Cantu-Paz, E. (1998). A Survey of Parallel Genetic Algorithms. *Calculateurs Paralleles, Reseaux et Systems Repartis,* 10, 141-171.

Cela, E. (1998). *The Quadratic Assignment Problem: Theory and Algorithms*, Kluwer Academic Publishers, Dordrecht.

Connoly, D. T. (1990). An improved annealing scheme for the QAP. *European Journal of Operational Research*, 46, 93-100.

Cung V.-D.; Mautor T.; Michelon P. & Tavares A. (1997). A Scatter Search based Approach for the Quadratic Assignment Problem. *Proceedings of the IEEE International Conference on Evolutionary Computation and Evolutionary Programming (ICEC'97)*, Indianapolis, 165-170.

de Carvalho Jr., S.A. & S. Rahmann (2006). Microarray Layout as a Quadratic Assignment Problem. *Proceedings of the German Conference on Bioinformatics, Lecture Notes in Informatics*. Vol. P-83, , 11-20, D. H. Huson; O. Kohlbacher; A. Lupas; K. Nieselt & A. Zell (eds.).

Drezner Z. (2002). Heuristic Algorithms for the Solution of the Quadratic Assignment Problem. *Journal of Applied Mathematics and Decision Sciences*, 6, 163-173.

Drezner Z. (2003). A New Genetic Algorithm for the Quadratic Assignment Problem. *INFORMS Journal on Computing*, 15, 320-330.

Drezner, Z. (2005a). Compounded Genetic Algorithms. *Operations Research Letters*, 33, 475-480.

Drezner Z. (2005b). A Distance Based Rule for Removing Population Members in Genetic Algorithms, *4OR*, 3, 109-116.

Drezner Z. (2005c). The Extended Concentric Tabu for the Quadratic Assignment Problem. *European Journal of Operational Research*. 160, 416-422.

Drezner Z. (2008a). Extensive Experiments with Hybrid Genetic Algorithms for the Solution of the Quadratic Assignment Problem. *Computers and Operations Research*, 35, 717-736.

Drezner Z. (2008b). Random Selection from a Stream of Events. *Communications of the ACM*. In press.

Drezner T. & Z. Drezner (2005). Genetic Algorithms: Mimicking Evolution and Natural Selection in Optimization Models. in *Biomimetics - Biologically Inspired Technologies,* Y. Bar-Cohen (Ed.), CRC Press, Boca Raton, FL, 157-175.

Drezner T. & Z. Drezner (2006). Gender-Specific Genetic Algorithms. *INFOR - Information Systems and Operations Research*, 44, 117-127.

Drezner Z.; P.M. Hahn & E.D. Taillard (2005). Recent Advances for the Quadratic Assignment Problem with Special Emphasis on Instances that are Difficult for Meta-heuristic Methods. *Annals of Operations Research*, 139, 65–94.

Drezner Z. & G.A. Marcoulides (2003). A Distance-Based Selection of Parents in Genetic Algorithms, in *Metaheuristics: Computer Decision-Making*, Mauricio G. C. Resende & Jorge P. de Sousa, (Eds.), 257-278, Kluwer Academic Publishers.

Drezner Z. & G.A. Marcoulides (2008). On the Range of Tabu Tenure in Solving Quadratic Assignment Problems. Under review. Also presented at the 4th International Conference on Computer Science and Information Systems, Athens, Greece, July, 2008.

Drezner Z. & S. Salhi (2002). Using Hybrid Metaheuristics for the One-Way and Two-way Network Design Problem. *Naval Research Logistics*, 49, 449-463.

Drezner Z. & G.O. Wesolowsky (1997). Selecting an Optimum Configuration of One-Way and Two-Way Routes. *Transportation Science*, 31, 386-394.

Drezner Z. & G.O. Wesolowsky (2003). Network Design: Selection and Design of Links and Facility Location. *Transportation Research Part A*, 37, 241-256.

Edmands S. (1999). Heterosis and Outbreeding Depression in Interpopulation Crosses Spanning a Wide Range of Divergence. *Evolution*, 53, 1757-1768.

Edmands S. (2002) Does Parental Divergence Predict Reproductive Compatibility? *Trends in Ecology & Evolution 17*, 520-527.

Epelman M.; S. Pollock; B. Netter & B. Low (2005). Anisogamy, Expenditure of Reproductive Effort, and the Optimality of Having Two Sexes. *Operations Research*, 53, 560-567.

Fleurent, C. & J.A. Ferland (1994). Genetic Hybrids for the Quadratic Assignment Problem. in *Quadratic Assignment and Related Problems* P. Pardalos & H. Wolkowicz, editors, , vol. 16, pp. 173-187. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*.

Gambardella, L.; Taillard, E. & Dorigo, M. (1999). Ant Colonies for the Quadratic Assignment Problem. *Journal of the Operational Research Society*, 50, 167-176.

Glover F. (1986) Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13, 533–549.

Glover F. & Laguna M. (1997) *Tabu search*. Kluwer, Boston.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Wokingham, England.

Hahn, P.M. & Krarup, J. (2001). A Hospital Facility Problem Finally Solved. *The Journal of Intelligent Manufacturing*, 12, 487-496.

Hansen, P. & Mladenovic, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130, 449–467.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor.

Koopmans T.C. & M. J. Beckmann (1957). Assignment problems and the location of economics activities. *Econometrica*, 25, 53-76.

Li, Y.; Pardalos, P.M. & Resende, M. (1994). A Descent Randomized Adaptive Search Procedure for the Quadratic Assignment Problem. in P. Pardalos and H. Wolkowicz (Eds.), *Quadratic Assignment and Related Problems*, Vol. 16, pages 237-261. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*.

Misevicius A. (2003). A Modified Simulated Annealing Algorithm for the Quadratic Assignment Problem. *Informatica*, 14, 1-18.

Misevicius A. (2003a). Ruin and recreate principle based approach for the quadratic assignment problem. Lecture notes in computer science, vol 2723. In: Cant-Paz E, Foster JA, Deb K, et al (eds) Genetic and evolutionary computation—GECCO 2003 (Chicago, USA), Proceedings, Part I, pp 598–609. Springer, Berlin Heidelberg New York.

Misevicius A. (2003b). Genetic algorithm hybridized with ruin and recreate procedure: application to the quadratic assignment problem. Knowl Based Syst 16:261–268.

Misevicius A. (2004). An improved hybrid genetic algorithm: new results for the quadratic assignment problem. Knowl Based Syst 17:65–73.

Misevicius A. (2005). A tabu search algorithm for the quadratic assignment problem. Working paper, Kaunas University of Technology, Kaunas, Lithuania Comput Optim Appl 30:95–111.

Mladenovic, N. & P. Hansen (1997). Variable neighborhood search. Computers and Operations Research 24, 1097– 1100.

Moscato, P. (2002). Memetic Algorithms. in *Handbook of Applied Optimization* P.M. Pardalos & M.G.C. Resende (Eds.), Oxford University Press, Oxford, U.K.

Nystrom M. (1999). Solving certain large instances of the quadratic assignment problem: Steinberg's examples. Working paper, California Institute of Technology, 1999.

Oliveira, C.A.S.; P.M. Pardalos & M.G.C. Resende (2004). GRASP with Path-relinking for the Quadratic Assignment Problem. In *Efficient and Experimental Algorithms*, LNCS, 3059, Ribeiro, C.C. and Martins, S.L. (eds.), 356-368, Springer-Verlag.

Pelikan M.; S. Tsutsui & R. Kalapala (2007). Dependency Trees, Permutations, and Quadratic Assignment Problem. Missouiri Estimation of Distribution Algorithms Laboratory (MEDAL) Report No. 2007003, University of Missouri in St. Louis, MO., January, 2007.

Rendl, F. (2002). The Quadratic Assignment Problem. in *Facility Location: Applications and Theory* Z. Drezner and H. Hamacher (Eds.) Springer, Berlin.

Rodriguez, J.M.; F.C. MacPhee; D.J. Bonham; J.D. Horton & V.C. Bhavsar (2004). Best Permutations for the Dynamic Plant Layout Problem. In *Proceedings of the 12th International Conference on Advances in Computing and Communications (ADCOM 2004)*. Dasgupta, A.R., Iyengar, S.S., and Bhatt, H.S. (Eds.), 173-178, Allied Publishers Pvt. Ltd., New Delhi Ahmedabad, India.

Skorin-Kapov, J. (1990). Tabu Search Applied to the Quadratic Assignment Problem. *ORSA Journal on Computing*, 2, 33-45.

Spears W.M. (2000). *The Role of Mutation and Recombination*, Natural Computing Series, Springer-Verlag.

Steinberg, L. (1961). The Backboard Wiring Problem: a Placement Algorithm. *SIAM Review*, 3, 37-50.

Taillard, E.D. (1991). Robust Tabu Search for the Quadratic Assignment Problem. *Parallel Computing*, 17, 443-455.

Taillard, E.D. (1995). Comparison of Iterative Searches for the Quadratic Assignment Problem. *Location Science*, 3, 87-105.

Tate, D.M. & A.E. Smith (1995). A Genetic Approach to the Quadratic Assignment Problem. *Computers and Operations Research*, 22, 73-83.

Thonemann, U.W. & A. Bolte (1994). An Improved Simulated Annealing algorithm for the Quadratic Assignment Problem. Working paper, School of Business, Department of Production and Operations Research, University of Paderborn, Germany.

Wilhelm, M.R. & T. L. Ward (1987). Solving Quadratic Assignment Problems by Simulated annealing. *IIE Transactions*, 19, 107-119.

6

A Hybrid GA-TS Technique with Dynamic Operators and its Application to Channel Equalization and Fiber Tracking

Luis M. San José-Revuelta
University of Valladolid
Spain

1. Chapter description

This chapter is based on the author's research activities during the last decade on the fields of Evolutionary Computation (EC)-based techniques applied to digital communications and medical image processing. Specifically, the chapter is organized in three main sections:

- The first section (section 2) presents a concise introduction to metaheuristic EC-based strategies, mainly genetic algorithms (GA) and tabu search (TS) - for the sake of comparison, brief comments to simulated annealing (SA) are included, as well. Section 2.1 shows a general description of the standard GA while section 2.2 focuses on the basic TS algorithm.
- Next, section 3 develops the proposed hybrid GA-TS method. It begins with the description of a genetic algorithm with notable reduced complexity (known as a *micro genetic algorithm*, μGA) that uses a modification of the standard genetic operators in order to improve its convergence rate and computational load. Such features are achieved by on-line tuning up the probabilities of mutation and crossover by means of analysing the population individuals' fitness entropy. This way, a new method to control and adjust the diversity of the population is obtained. The μGA here described was partially developed in (San José, 2005). Once the GA is obtained, it is then modified and improved using the main distinctive concepts of TS. Specifically, we introduce a systematic use of memory in order to keep information on the last visited solutions as well as on the concrete parts of the chromosomes, or population's individuals, that have experimented alterations that have positively or negatively affected the fitness function. Besides, memory keeps track of the genes affected by the genetic operators and the tabu tenure depends on the explorative or exploitative sense of the search, which is estimated from the mean population fitness entropy previously described. This way, the TS main ideas will help to avoid both cycling and processing of non-interesting regions of the solutions' space. The hybrid algorithm thus developed will be denoted as "GA-TS".
- The last part of the chapter is devoted to the description of two application examples:
  (1) Application of the GA-TS algorithm to symbol detection in synchronous wireless communications. Numerical results will analyze the performance in comparison to traditional methods such as the matched filter detector, the minimum mean square

error algorithm, the standard GA-based detector as well as some radial basis function (RBF)-based methods.
(2) Application of the developed GA-TS method to estimate the parameters of a recently developed algorithm (San José, 2007) for fiber tracking in diffusion tensor (DT) fields acquired via magnetic resonance imaging (MRI). This algorithm is successfully tested with both intricate synthetic images and real white matter DT-MR images. Numerical experiments will show the performance gain over previous approaches, especially with respect to convergence and computational load. Tracking of white matter fibers in the human brain becomes an issue of essential importance nowadays since it will improve the diagnosis and treatment of many neuronal diseases.

2. Introduction to metaheuristic EC-based strategies

2.1 Genetic algorithms

2.1.1 GA fundamentals

Genetic algorithms are a part of evolutionary computation (EC), which is a rapidly growing area of artificial intelligence. GAs are inspired by Darwin's theory about evolution. Simply said, these algorithms encode a potential solution to a specific problem on a simple data structure (known as *chromosome* or *individual*) and apply genetic operators to a selected group of the whole set of potential solutions (known as *population*) so as to preserve critical information. This is motivated by the hope that the new population will be better than the old one. Those chromosomes selected to form new individuals (*offspring*) are selected according to their *fitness* - the more suitable they are, the more chances they have to reproduce. Following the analogy with natural systems, the complete set of chromosomes is called *genome*. A particular set of genes in genome is called *genotype* (Mitchell, 1996).
Genetic algorithms are often viewed as function optimizers, although the range of problems to which they have been applied is quite broad. An implementation of a GA begins with a population of typically randomly generated chromosomes. These individuals are then evaluated and assigned reproductive opportunities so that those chromosomes representing a better solution to the target problem are given more chances to reproduce than those chromosomes which are poorer solutions.
Under this particular description of a GA, the term "genetic algorithm" has two meanings: In a strict interpretation, the GA refers to a model introduced and investigated by John Holland and his colleagues (Holland, 1975). Most of the existing theory for GAs applies to this model as well as variations on what is frequently referred to as the "standard genetic algorithm". In a broader usage of the term, a genetic algorithm is any population-based model that uses selection and recombination operators to generate new sample points in a multidimensional search space.
Considering an application oriented GA implementation the size of the search space is related to the number of bits used in the problem encoding, i.e. for a bit string encoding of length L, the size of the search space would be 2^L and it can be viewed as a hypercube. In this situation, the genetic algorithm can be considered as a method for sampling the corners of this L-dimensional hypercube. These type of problems are also known as "*NP-complete problems*", where *NP* stands for *nondeterministic polynomial* and it means that it is possible to "guess" the solution (by some nondeterministic algorithm) and then check it, both in polynomial time. A well-known example of NP problem is the *travelling salesman problem*.

Since the number of good solutions to a problem is sparse with respect to the size of the search space, then random search or search by enumeration becomes an impractical form of problem solving. Besides, any search other than random search imposes some bias in terms of how it looks for better solutions and where it looks in the search space. Though genetic algorithms indeed introduce a particular bias in terms of what new points in the space will be sampled, they make relatively few assumptions about the problem that is being solved. As a weak method, GAs are robust but very general. If there exists a good specialized optimization method for a specific problem, then genetic algorithm may not be the best optimization tool for that application. On the other hand, some researchers work with hybrid algorithms that combine existing methods with genetic algorithms. In this chapter we propose to enhance the performance of a particular GA (San José, 2005) with specific concepts representative of tabu search.
The theoretical concepts that explain how genetic algorithms work, i.e. how chromosomes evolve toward the optimum solution, are partially based on the *Schema Theorem*. For the sake of brevity, its description lies beyond the scope of this chapter.

2.1.2 GA cycle

In summary, a GA starts with a *population* or set of possible solutions represented by *chromosomes*. Solutions from one population are taken and used to form a new population. Solutions which are selected to form new solutions (*offspring*) are selected according to their fitness - the more suitable they are, the more chances they have to reproduce. This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied. Figure 1 outlines the main steps of the basic genetic algorithm.
This outline of basic GA is very general and there exist many things that can be implemented differently in various problems, for instance: how to create chromosomes, what type of encoding choose, how to define the two basic operators of GA (*crossover* and *mutation*), how to select parents for crossover, and many other implementation issues. Some of the concerning questions will be discussed in the specific applications later described.

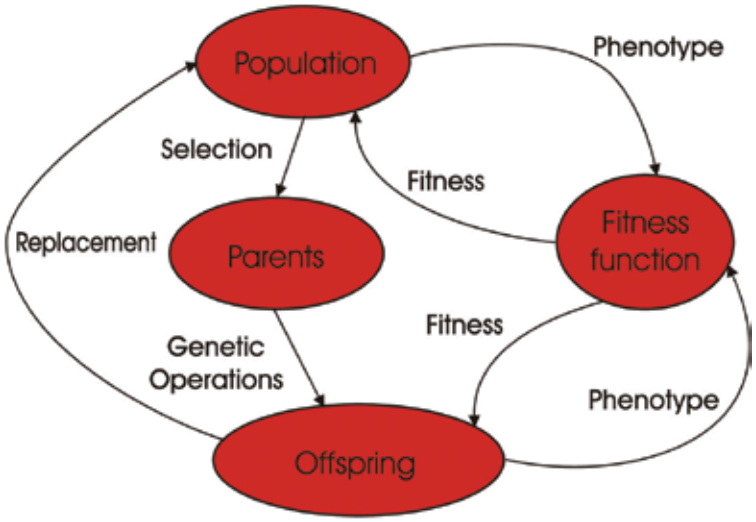


Fig. 1. Outline of the basic genetic algorithm.

2.1.3 Encoding and genetic operators

Crossover and mutation are the most important parts of a genetic algorithm. Performance is influenced by these two operators. However, the description of the chromosome encoding must be first commented.

The chromosome should contain information about the solution that it represents. The most used way of encoding is a binary string, for instance: $\mathbf{u}_1$=[1101100100110110], $\mathbf{u}_2$=[1101111000011110]. Each bit in this string can represent some characteristic of the solution or the whole string can represent a number. Other ways of encoding include directly integer or real numbers encoding.

Once encoding has been decided, the genetic operators (crossover and mutation) must be defined. Crossover selects genes from parent chromosomes and creates a new offspring. The simplest way to achieve this task is to choose randomly a few crossover points and interchange the genetic material separated by these points as illustrated in Figure 2.

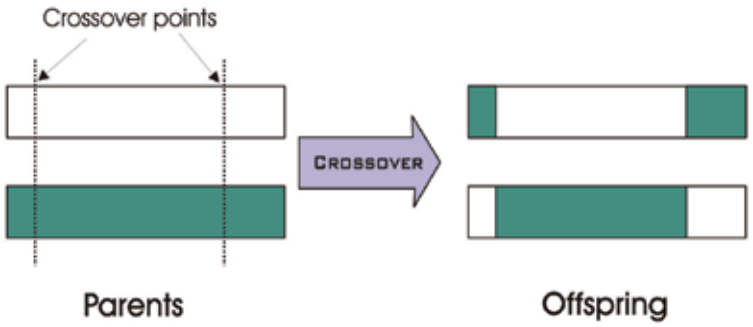


Fig. 2. Crossover example with two crossover points.

Notice that it is also possible to define a crossover operator with only one or even with multiple crossover points. Crossover can be rather complicated and depends on the encoding of the chromosome. Specific crossover made for a specific problem can improve performance of the genetic algorithm. For instance, in our applications, the *uniform crossover* and the *partially matched crossover* (PMX) operators will be used.

The crossover operator requires the definition of a selection procedure in order to select the parents (two chromosomes) from the population. According to Darwin's evolution theory, the best ones should survive and create new offspring. There are many methods for selecting the best chromosomes, for example roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others. In our specific applications, the *roulette wheel selection* scheme will be used. Using this selection procedure, the better the chromosomes are, the more chances to be selected they have.

Once crossover is performed, mutation takes place. Mutation changes randomly the new offspring. For binary encoding we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1. Obviously, both mutation and crossover depend on the encoding. An example of mutation is shown in Figure 3.

Consequently, there are two basic parameters of GA - the crossover probability and the mutation probability. Since we will propose a novel strategy for on-line adjusting these probabilities based on the entropy of the population individuals' fitness, a detailed description of these parameters as well as their influence on global convergence will be explained in sections 3.6-3.7.

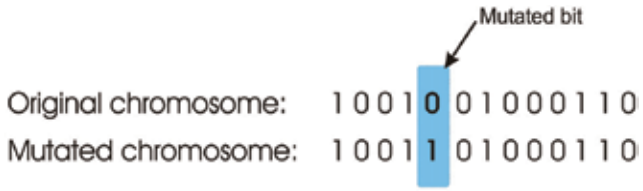


Fig. 3. Mutation example.

Thus, the GA works repeating a cycle: every iteration, those chromosomes with highest fitness are selected for creating a new offspring. Then, those ones with the lowest fitness are removed and the new offspring is placed in their place. The remaining chromosomes of the population survive to next generation.

2.1.4 Elitism and final remarks

Finally, we will comment the concept of *elitism* which is implemented in many GAs. This strategy was developed with the aim of avoiding the possibility of loosing the best chromosome in the current population when the genetic operators are applied. When elitism is used, the best chromosome (or a few best chromosomes) is directly copied into the new population. The rest is done in the same way. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution.
In summary, among the main advantages of GA we find: (i) their parallelism. GAs search the solutions' space with more individuals (and with genotype rather than phenotype) so they are less likely to get stuck in a suboptimal or local solution like some other methods. (ii) they are easy to implement. Once you have some GA, you just have to write the new chromosome (just one object) to solve a different problem. On the other hand, their main drawback is their computational load. If not properly designed, they can be slower than some other methods.

2.2 The Tabu search strategy

During the last decade, the tabu search (TS) technique has shown a remarkable efficiency on many problems. Tabu search was originally presented in its present form by Glover (Glover, 1986); the basic ideas have also been sketched by Hansen (Hansen, 1986). Additional efforts of formalization are reported in (Glover, 1989; de Werra & Hertz, 1989; Glover, 1990). Up to now, there is no formal explanation of this good behaviour, though, theoretical aspects of tabu search have been investigated (Faigle & Kern, 1992; Glover, 1992; Fox, 1993).
Tabu search belongs to the iterative techniques group, a type of optimization procedures. The general step of an iterative procedure consists in constructing from a current solution i a next solution j and in checking whether one should stop there or perform another iteration. Neighbourhood search methods are iterative procedures in which a neighbourhood $N(i)$ is defined for each feasible solution i, and the next solution j is searched among the solutions in $N(i)$. Simulated annealing (SA) and tabu search can be considered as neighbourhood search methods which are more elaborate than the classical descent method. The basic ingredients of tabu search are next described.

2.2.1 Fundamentals

In order to improve the efficiency of the search, it is necessary to keep track not only of local information (like the current value of the objective function) but also of some information

related to the exploration process. The use of *memory* is one of the essential features of TS. While most exploration methods keep in memory essentially the value $f(i^*)$ of the best solution i^* visited so far, TS also keeps information on the *itinerary* through the last solutions visited. Such information will be used to guide the next move. Memory restricts the possibilities to some subset of $N(i)$ by forbidding for instance moves to some neighbour solutions. Specifically, the structure of the neighbourhood $N(i)$ of a solution i will vary from iteration to iteration. This is why TS is included in a class of procedures called *dynamic neighbourhood search techniques.*

The formal description is as follows (Hertz, 1995): Let us consider an optimization problem in the following way: given a set S of feasible solutions and a function f: $S \rightarrow \Re$, find some solution i^* in S such that $f(i^*)$ is acceptable with respect to some criterion (or criteria). Generally a criterion of acceptability for a solution i^* would be to have $f(i^*) \leq f(i)$ for every I in S. In this situation, TS would be an exact minimization algorithm provided the exploration process would guarantee that after a finite number of steps such an i^* would be reached.

However, in most contexts, no guarantee can be given that such an i^* will be obtained. Therefore, TS could simply be viewed as a general heuristic method. Since TS includes in its own operating rules some heuristic techniques, it would be more appropriate to characterize TS as a *metaheuristic*. Frequently, its role will be to guide and to orient the search of another (more local) search procedure. As a first step towards the description of TS, the classical descent method can be formulated as follows:

Step 1. Choose an initial solution i in S.

Step 2. Generate a subset V^* of solution in neighbourhood $N(i)$.

Step 3. Find the best j in V* ($f(j) \leq f(k)\ \forall k \in V^*$) and set $i=j$.

Step 4. If $f(j) \geq f(i)$ then stop. Else go to Step 2.

A descent method would generally take $V^*=N(i)$. Since this approach may often be too time-consuming, an appropriate choice of V^* may be a notable improvement.

The opposite case would be to take $|V^*|=1$; this would drop the phase of choice of a best j. Simulated annealing could be integrated within such a framework. A solution j would be accepted if $f(j) \leq f(i)$, otherwise it would be accepted with a certain probability depending upon the values of f at i and j as well as upon a parameter called *temperature* (there is no temperature in TS). Memory is used to choose V^* so as to exploit knowledge extending beyond the function f and the neighbourhood $N(i)$.

Descent procedures often get trapped in a local minimum. So any iterative exploration process should in some instances accept also non-improving moves from i to j in V^* (i.e. with $f(j) > f(i)$) if one would like to escape from a local minimum. Simulated annealing does this also, but it does not guide the choice of j, TS in contrast chooses a best j in V^*. However, this strategy increases the risk of cycling and re-visiting solutions. This is the point where memory is helpful to forbid moves which might lead to recently visited solutions. This way, the structure of $N(i)$ will depend upon the itinerary and hence upon the iteration k, so we may refer to $N(i,k)$ instead of $N(i)$. Taking this idea into account, the descent algorithm can be reformulated in a way which will bring it closer to the general TS procedure. It can be stated as follows (Hertz, 1995):

Step 1. Choose an initial solution i in S. Set $i^*=i$ and $k=0$.

Step 2. Set $k=k+1$ and generate a subset V^* of solution in $N(i,k)$

Step 3.Choose a best j in V^* (with respect to f or to some modified function $\tilde{f}$ and set $i=j$.

Step 4. If $f(i) < f(i^*)$ then set $i^*=i$.

Step 5. If a stopping condition is met, then stop. Else, go to Step 2.

Notice that we may consider the use of a modified $\tilde{f}$ instead of f in some circumstances. Some of the stopping conditions are the following: (i) $N(i,k+1)$ is empty, (ii) k is larger than the maximum number of iterations allowed, (iii) the number of iterations since the last improvement of i^* is larger than a specified threshold, and (iv) there exists evidence than an optimum solution has been obtained.

The definition of $N(i,k)$ at each iteration k and the choice of V^* are crucial for the final search result. The first one implies that some recently visited solutions are removed from $N(i)$. They are considered as *tabu solutions* which should be avoided in the next iteration. Such a memory based on recency will partially prevent cycling. Specifically, keeping at iteration k a list T (tabu list) of the last $|T|$ solutions visited will prevent cycles of size at most $|T|$. In such a case $N(i,k)=N(i)-T$. Since this list T may be extremely impractical to use, we will describe the exploration process in terms of moves from one solution to the next. For each solution i in S, we define $M(i)$ as the set of moves which can be applied to i in order to obtain a new solution j (notation: $j=i\oplus m$). Then $N(i)=\{j \;/\; \exists\; m\in M(i)$ with $j=i\oplus m\}$. So, instead of keeping a list T of the last $|T|$ solutions visited, we may simply keep track of the last $|T|$ moves. Obviously, this restriction losses information and that it does not guarantee that no cycle of length at most $|T|$ will occur. For efficiency purposes, it may be convenient to use several lists T_r at a time. Then some constituents t_r (of i or of m) will be given a tabu status to indicate that these constituents are currently not allowed. Generally, the tabu status of a move is a function of the tabu status of its constituents which may change at each iteration. A collection of tabu conditions can be formulated as

$$t_r(i,m) \in T_r(r = 1,...,t). \tag{1}$$

A move m will be a tabu move if *all* conditions are satisfied. A drawback of the replacement of solutions by moves is that unvisited solutions can be given a tabu status. We will then overrule the tabu status when some tabu solutions will look attractive. This is performed by means of *aspiration level conditions*.

A tabu move m applied to a current solution i may appear attractive because it gives, for example, a solution better than the best found so far. We would like to accept m in spite of its status; we shall do so if it has an *aspiration level* $a(i,m)$ which is better than a threshold value $A(i,m)$. Parameter $A(i,m)$ can be viewed as a set of preferred values for a function $a(i,m)$. Thus, conditions of aspiration can be written in the form

$$a_r(i,m) \in A_r(i,m)\ (r=1,...,a). \tag{2}$$

If at least one of these conditions is satisfied, then m will be accepted.

Finally, in some situations, it can be of interest to replace the process f by another function $\tilde{f}$ so as to introduce some intensification and diversification of the search. In the search process it is sometimes fruitful to intensify the search in some region of S because it may contain some acceptable solutions (*exploitative* search). Such intensification can be carried out by giving a high priority to the solutions which have common features with the current

solution. This can be done with the introduction of an additional term in the objective function that penalizes solutions far from the present one. This should be done during a few iterations and after this it may be useful to explore another region of *S*. On the other hand, diversification (*explorative* search) can be forced by introducing an additional term in the objective function in order to penalize solutions that are close to the present one. The modified objective function is the function $\tilde{f}$ that was mentioned earlier in the algorithm:

$\tilde{f} = f + Intensification + Diversification$.

Gathering together all these concepts, the TS procedure can be finally stated as follows (Hertz, 1995):

Step 1. Choose an initial solution *i* in *S*. Set $i^*=i$ and $k=0$.

Step 2. Set $k=k+1$ and generate a subset V^* of solution in $N(i,k)$ such that either one of the tabu conditions $t_r(i,m) \in T_r$ is violated ($r=1,...,t$) or at least one of the aspiration conditions $a_r(i,m) \in A_r(i,m)$ holds ($r=1,...,a$).

Step 3. Choose best $j=i \oplus m$ in V^* (with respect to *f* or to the function $\tilde{f}$) and set $i=j$.

Step 4. If $f(i) < f(i^*)$ then set $i^*=i$.

Step 5. Update tabu and aspiration conditions.

Step 6. If a stopping condition is met, then stop. Else, go to Step 2.

3. Proposed hybrid GA-TS algorithm

This section describes a hybrid GA-TS algorithm whose main novelty comes from its very low computational load (similar to a μGA), the introduction of an on-line procedure to adjust the GA's parameters in order to achieve and maintain a good population diversity, and the incorporation of memory concepts proper of TS that improve convergence speed and diversity, avoiding cycling and reducing computational load. This diversity is a key issue in the performance of any evolutionary algorithm, including GAs and TS methods (Ursem, 2002). For this reason, we first briefly comment the importance of diversity in relation to convergence properties, and, afterwards, we will present the specific GA-TS structure together with the diversity control algorithm.

3.1 Diversity and suboptimal convergence in evolutionary computation

As mentioned in section 2, both genetic algorithms and tabu search constitute robust global search and optimization strategies that can strike an attractive balance between the *exploitation* - vertical search in the proximities of an specified point or area in the solutions space - and the *exploration* - horizontal search throughout all the totality of the solutions space, allowing the test of new potential solutions - of possible problem solutions. However, simple GAs have a tendency to converge prematurely to local optima, mainly due to selection pressure and too high gene flow between population members (Ursem, 2002). First, a high selection pressure will fill the population with clones of the best fit individuals, since they have the highest survival probability. Diversity declines after a short while, and, because the population consists of similar individuals, the algorithm will have difficulties escaping the local optimum. Nonetheless, lowering the selection pressure will often lead to an unacceptable slow convergence speed. On the other hand, high gene flow is often determined by the population structure. In simple GAs any individual can mate with any

other individual. Therefore, genes spread fast throughout the population and the diversity drops quickly with fitness stagnation as a prevalent outcome.
All these facts point out the key role of maintaining a suitable diversity in the population in order to appropriately converge to the optimal solution, thus avoiding, the inefficient presence of duplicated or very similar individuals, as well as the possibility of getting trapped into suboptimal solutions. Some authors have already dealt with the problem of diversity monitoring and control: For instance, the Diversity-Control-Oriented GA (Shimodaira, 1999) calculates a survival probability by means of a diversity measure based on the Hamming distance between the individual and the current best individual. Hence, diversity is preserved through the selection procedure. Another approach is the Shifting-Balance GA (Oppacher, 1999), where a containment factor between two subpopulations - based on Hamming distances between all members of the two populations - determines the ratio between individuals selected on fitness and individuals selected to increase the distance between the two populations. A third, and more distantly related, approach is the Forking GA (Tsutsui, 1997), which uses specialized diversity measures to turn a subset of the population into a subpopulation. Two variants of the Forking GA exist. The first one operates on the genotype, whereas the second type bases the division on distances in the search space (on the phenotype). More recently, Ghosh et al. (Ghosh, 2003) proposed to vary the mutation probability in n equally-length intervals along the n_g iterations of the algorithm, while the probability of crossover was kept constant (see Fig. 3 in (Ghosh, 2003), p. 863, for an example). The authors remark that this scheme increases the diversity of the population when the probability of mutation is increased and, conversely, as the optimal string is approached, the probability of mutation is reduced.

3.2 Fundamentals and encoding

The principle of GAs consists in representing a set of potential solutions (*population*) with a predetermined encoding rule. Each potential solution (*chromosome*) is associated to a figure of merit, or *fitness* value, in accordance to its proximity to the optimal solution, i.e., each chromosome is evaluated for its fitness in solving a given optimization task. When no prior knowledge of the solution is available, this initial set of potential solutions, $P[0]$, which forms the whole population for each new signaling interval (iteration k=0), is randomly generated. We will denote $P[k] = \{\mathbf{u}_i\}_{i=0}^{n_p}$ to the population at iteration k, with n_p being the number of individuals $\mathbf{u}_i$ per generation. For instance, the specific encoding scheme used for the first application later considered is explained in section 4.3.

3.3 Genetic operators

Once individuals are generated and given a fitness value, the next step consists in applying the *genetic operators*, mainly *mutation* and *crossover*. The first one modifies specific individuals with probability p_m, changing (antipodal bit) the value of some concrete position/s in the encoding of $\mathbf{u}_i$. Both the position and the new value are randomly generated - see Fig. 2. A low level of mutation serves to prevent any element in the chromosome from remaining fixed to a single value in the entire population. However, a high level of mutation will essentially result in a random search. Hence, the value of p_m must be chosen carefully in order to avoid excessive mutation. To maintain a satisfactory balance between such extremes a good initial value for p_m in our applications is between 0.01 - 0.05.

Note that mutation promotes exploration by creating an opportunity to explore different areas of the solution space. A good element can even turn into a bad one. This is definitely undesirable, especially if the offspring, which constitutes the actual global optimum, is mutated to a suboptimum solution. Hence, the value of p_m must be carefully chosen in order to prevent excessive mutation.
On the other hand, the crossover operator requires two operands (*parents*) to produce two new individuals (*descendants* or *offspring*). These new individuals are created when merging parents by crossing them at specific internal points - see Fig. 3. This operation is performed as follows: parent individuals $\mathbf{u}_i$ and $\mathbf{u}_j$ are exchanged using the *uniform crossover* process (Mitchell, 1996) in order to produce two offspring individuals. The process of uniform crossover uses a so-called *crossover mask*, which is a sequence of randomly generated 0's and 1's. The elements in $\mathbf{u}_i$ and $\mathbf{u}_j$ are exchanged at bit locations corresponding to a "1" in the crossover mask with probability p_c.

3.4 Elitism and termination criteria

Since parents are much more likely to be selected from those individuals having a higher fitness, the small variations introduced within these individuals are intended to also generate high fit individuals. Using Markov chain modelling, it has been proved that Gas are guaranteed to asymptotically converge to the global optimum - with any choice of the initial population - if an elitist strategy is used, where at least the best chromosome at each generation is always maintained in the population (Bhandari, 1996). However, Bhandari et al. (Bhandari, 1996) provided the proof that no finite stopping time can guarantee the optimal solution, though, in practice, the GA process must terminate after a finite number of iterations with a high probability that the process has achieved the global optimal solution.
In the proposed GA-TS algorithm, the elite E_k for $P[k+1]$ is formed by selecting those individuals from both the elite of $P[k]$ and the mutated elite of $P[k]$ having the highest objective value in the population. The mutation of the elite is performed with a probability $p_{m,e}$ considerably smaller than p_m (so as to avoid the destruction of good solution guesses). In our simulations we used $p_{m,e} = x\ p_m$, with x=0.2 (heuristical trials show little differences for values of x in the range [0.1,0.5]). No crossover is performed on the elite, since it implies, in most of cases, abandoning the exploitation of these good potential solutions.
This procedure, whose flowchart is shown in Fig. 4, is iterated a predefined number of consecutive generations, n_g. At the end, the string $\mathbf{u}_i$ corresponding to the best fit individual is finally chosen as the problem solution.

3.5 Diversity and entropy-dependent genetic operators

Standard GAs suffer from an excessive computational load: the application of the genetic operators is often costly and the evaluation of fitness can also be a very time-consuming task. Populations sizes n_p normally are 100, 200 or even much higher - for instance, (Uursem, 2002) uses populations with 400 individuals.
Hence, we propose an algorithm works with much smaller population sizes (in the order of 15 to 35 individuals). An elite of 2 to 5 individuals is kept and the crossover and mutation probabilities depend on the Shannon entropy of the population (excluding the elite) fitness which is calculated as

$$H(P[k]) = -\sum_{i=1}^{n_p} \lambda_i(k) \log \lambda_i(k) \tag{3}$$

with λ_i being the normalized fitness of individual $\mathbf{u}_i$. When all the fitness values are very similar, with small dispersion, $H(P[k])$ becomes high and p_c is decreased (it is not worthwhile wasting time merging very similar individuals). This way, the exploration character of the search is boosted, while, conversely, exploitation decreases. On the other hand, when this entropy is small, there exists a high diversity within the population, a fact that can be exploited in order to increase the horizontal sense of search. Following a similar reasoning, the probability of mutation is increased when the entropy is high, so as to augment the diversity of the population and escape from local suboptimal solutions (exploration decreases, exploitation becomes higher).

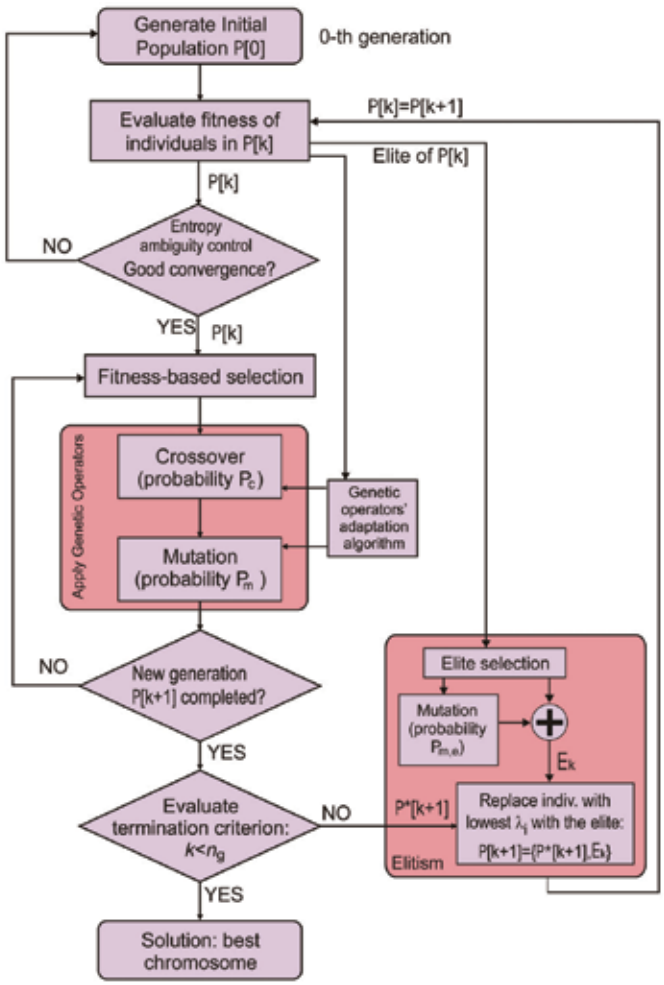


Fig. 4. Schematic representation of the GA structure and the genetic operators.

Therefore, we have that probabilities p_m and p_c are directly/inverselly proportional to the population fitness entropy, respectively[1],

[1] Symbol " α " denotes proportionality.

$$p_c(k) \propto [H(P[k])]^{-1} \tag{4}$$

$$p_m(k) \propto H(P[k]) \tag{5}$$

Some exponentially dependence on time k is also included in the model - making use of exponential functions - in order to relax (decrease), along time, the degree of dependence of the genetic operators' probabilities with the dispersion measure[2]. This avoids abandoning good solution estimates when very noisy sporadic samples are received or when the whole population has converged to the global optimum.
Hence, the proposed algorithm uses a much smaller population size than standard GAs, calculates crossover with a very low probability (and only on individuals not belonging to the elite), and the exploration/exploitation sense of the search is on-line tuned up by measuring the diversity of the population by means of its fitness entropy.
Next section describes how the diversity of the population affects the genetic operators' probabilities.

3.6 Genetic operators and convergence cycle

A typical estimation convergence process is depicted in Fig. 5. Notice that this is just one - the most frequently found and representative - of the possible situations, which will greatly depend on the randomly generated initial population $P[0]$ and the specific application. This is also a mere schematic representation and the vertical axis should have a different scale for each represented parameter.

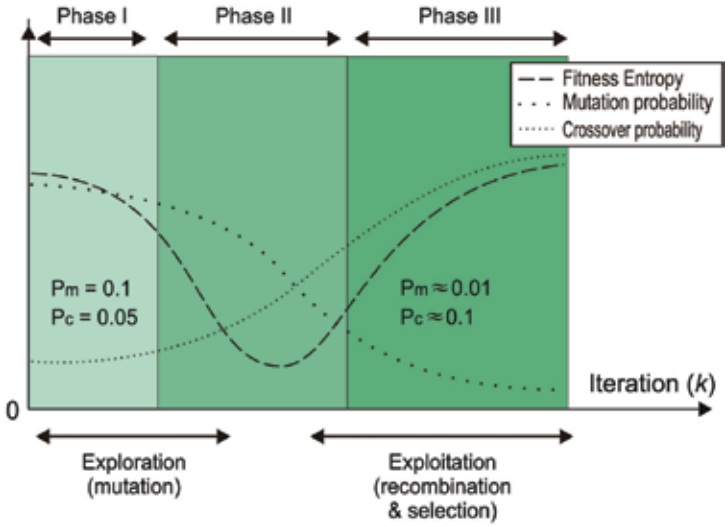


Fig. 5. Schematic representation of the alternating exploring and exploiting behavior in the diversity-guided GA-TS. The proportionality between different curves is not represented; only the increasing/decreasing tendency of each parameter is representative.

[2] See, also, complementary discussion on the form of these two functions, p_c and p_m, in section 3.7 *Genetic operators and convergence cycle*.

Three different phases can be appreciated:

(I) Population $P[0]$ has been randomly generated. Individuals $\mathbf{u}_i$ share a very few bits I common. Their fitness values are very similar since all of them are, probably, wrong solution estimates and, thus, the entropy is high. The probability of mutation is kept high so as to explore new potential solutions and the probability of crossover is small.

(II) Some individuals begin to converge to good solution estimates. Their fitness increase while the remaining individuals of the population continue presenting low values. The entropy of the fitness considerably decreases. The probability of mutation is decreased gradually while the one associated to crossover becomes higher so as to exploit the genetic information of the good estimates.

(III) Most of the population individuals have converged to good solutions. The fitness of all the individuals is very close (all of them tend to be equiprobable solutions). The entropy is again high and the exploitative behavior prevails over the explorative one.

As an example, since concrete values greatly depend on the randomly generated initial population, Fig. 5 also shows the approximate range in which p_c and p_m evolve along the execution of a satisfactory-convergence typical cycle: $p_m(I)=0.1 \rightarrow p_m(III)\approx 0.01$ and $p_c(I) = 0.05 \rightarrow p_c(III)\approx 0.1$.

Keeping these three different phases in mind, the determination of the form of functions p_c and p_m in Eqs. (4) and (5) is achieved as follows:

- **Probability of mutation**: during phase I, where most of the individuals are wrong estimates and the entropy is high, p_c is required to be high to as to increase the explorative sense of the search. Thus, p_c is directly proportional to the entropy H and to another function, which is denoted as g_m, and has the form $g_m(k) = \exp(-\beta_m k)$, which, for low values of k and $\beta_m < 1$, is close to 1. Thus, we can write

$$p_m(k) = \xi_m \cdot g_m(k) \cdot H(P[k]) \tag{6}$$

  where ξ_m stands for a normalization parameter, which ensures that $p_m \in [0,1]$, and $g_m(k) = \exp(-\beta_m k)$ with $0 < \beta_m < 1$.

  Once the GA-TS algorithm has converged to some good solution estimates, H becomes high again (see Phase III in Fig. 5), but now p_m must be lowered so as to switch from exploration to the exploitation of new individuals. This is achieved selecting g_m such that its decreasing character prevails over the entropy values. In our particular system, exponential functions $g_m(k) = \exp(-\beta_m k)$, with $\beta_m \in [0.1 , 0.05]$, were used.

- **Probability of crossover**: in this case, p_c should maintain low values in Phase I in order to allow an explorative search. In this phase H is high and p_c is chosen to be inversely proportional:

$$p_c(k) = \frac{\xi_c g_c(k)}{H(P[k])} \tag{7}$$

  If function g_c has the form of an inverse exponential (logarithmic) function, durin Phase I, the product of g_c and the inverse of H adopts low values (see Phase I in Fig. 5), while in the third phase the g_c prevails over the low values of the inverse of the entropy, leading to high probabilities (Phase III), in accordance to the desired exploitative sense of the search.

In both cases, probabilities must be properly normalized in the range [0,1] making use of the parameters ξ_m and ξ_c. The weight of this updating of both probabilities also decreases with time, so that when Phase II is widely surpassed, the mutation and crossover probabilities remain mainly constant.

3.7 Eliminating the entropy ambiguity

The use of the entropy function as defined in Eq. (3) with the aim of classifying populations, can lead, if not properly corrected, to ambiguous situations since a population with all its individuals being very similar and having high fitness values will show the same high entropy as another population with very similar individuals taking on, all of them, very low fitness values. Obviously, the dispersion in both cases is low, and the associated entropy will show a high value. In order to detect and differentiate these situations, the following simple strategy has been devised.

Once a few generations have been processed (about 10–15), the following parameters are compared:

$$\Gamma_1^{\alpha}(k) = \frac{1}{\alpha}\sum_{j=0}^{\alpha-1}\left[\frac{1}{n_p}\sum_{i=1}^{n_p}\lambda_i(k-j)\right] \tag{8}$$

$$\Gamma_0^{\alpha}(k) = \frac{1}{\alpha}\sum_{j=0}^{\alpha-1}\left[\frac{1}{n_p}\sum_{i=1}^{n_p}\lambda_i(k-(j+\zeta))\right] \tag{9}$$

The first one, $\Gamma_1^{\alpha}(k)$, represents the averaged mean value of the individuals' fitness in a successive generations –the current one, k, and the α −1 generations before. The second parameter, $\Gamma_0^{\alpha}(k)$, represents this same value but averaging the mean fitness values in a group of α iterations, ζ generations before. For instance, the simulations for the first application proposed in this chapter were carried out with α =3 and ζ=8, thus the mean value of the fitness values averaged over the last 3 successive iterations is compared to this value 8 generations before.

For example, at generation k=11, the algorithm compares the averaged value of the mean fitness in generations 9, 10 and 11, with the averaged value of the fitness in the generations 1, 2 and 3.

In cases of good convergence (or, at least, when the GA-TS algorithm has begun to converge), the difference between $\Gamma_1^{\alpha}(k)$ and $\Gamma_0^{\alpha}(k)$, will be higher than a predetermined threshold Γ_{th} (heuristically determined). On the other hand, when the population contains low fitness individuals after $2\alpha+\zeta$ generations, it is a clear indicator of unsatisfactory convergence. In this case the whole population is randomly re-initialized.

According to this, the difference between both averaged values, is compared to a threshold Γ_{th}. Whenever condition

$$\Gamma_1^{\alpha}(k) - \Gamma_0^{\alpha}(k) < \Gamma_{th} \tag{10}$$

is satisfied, the whole population will be randomly re-initialized.

3.8 GA improvement with TS concepts

The search algorithm described before has been improved by incorporating some of the most important and characteristic concepts of TS. Specifically, *memory* has been implemented so as to avoid revisiting solutions. Depending on the application, *tabu conditions* t_r have been defined representing non-possible or incongruent solutions. This mechanism, together with an appropriate *diversity control* based on the entropy dependent operators leads to a very efficient hybrid GA-TS method that avoids cycling search, improves convergence (as a consequence of improving diversity) and can be implemented in a computationally efficient algorithm.

The hybrid GA-TS thus implemented can be viewed as an advanced TS method with multiple hypotheses i evolving in parallel and with a powerful procedure to evolve to the next iteration using the dynamic operators here developed. This method can perfectly be classified as a hybrid evolutionary algorithm combining the advantages of both GAs and TS.

4. Application I: multiuser detection for DS/CDMA communications

This section shows the application of the previously described GA-TS algorithm to the problem of symbol detection in DS/CDMA multiuser communications. The thus obtained detector has an extremely low computational load and offers an interesting alternative to previous suboptimal algorithms whose performance is frequently subject to the near-far problem and multiple access interference degradations. Its performance is compared to that of standard GA-based detectors, as well as traditional multiuser detectors, such as the matched filter, the decorrelator and the MMSE detectors. This section is mainly based on (San José, 2005).

4.1 Problem description

During the last decade, the utilization of wireless communications has shown growth rates of 20-50% per year in various parts of the world. The use of Code-Division Multiple Access (CDMA) communications has received a considerable amount of attention as mobile cellular telephone providers look for schemes of transmission which can exploit the capacity of the available spectrum to the maximum (Glisic, 1997; Proakis, 1995; Verdú, 1998)

Since CDMA systems give users access to very high data rates, intersymbol interference (ISI) cannot be neglected and, together with multi-access interference (MAI), constitutes the major drawback to the overall system performance (Proakis, 1995). Both effects, if not appropriately controlled, can seriously deteriorate the quality of reception.

Numerous methods have been proposed for reducing the amount of MAI present in the received signal, such as power control, optimization of signature sequences or sectorized antennas. Conventional receivers, which rely on a filter matched to the signature of the user of interest, are only optimum when the set of received signatures is perfectly orthogonal, which is not the most common case found in practice. Thus, their performance is notably degraded, especially when near-far effects exist. In 1986, Verdú showed that this problem could be solved by jointly-extracting the information sequences of all the users (Verdú, 1998). Unfortunately, the complexity of the optimum multiuser detector (MUD) based on the maximum likelihood (ML) rule increases exponentially with the number of active users, making it impractical for realistic environments. Consequently, the development of suboptimal detectors has deserved a considerable amount of attention during last years.

Many of these suboptimal schemes make use of Natural Computation techniques, such as genetic algorithms and neural networks (Ergun, 2000; Juntti, 1997; Shayesteh, 2001; Shayesteh, 2003; Wang, 1998; Yen, 2000).
A GA-based MUD was first proposed by Juntti et al. in (Juntti, 1997). The analysis is based on a synchronous CDMA system and requires good initial guesses concerning the possible user symbol sequences obtained from the other detectors. However, Yen et al. (Yen, 2000) showed that, by incorporating an element of local search prior to invoking the GA, the performance approaches the single-user performance bound. The proposal by Ergün et al. (Ergun, 2000) uses a multistage MUD as part of the GA-aided detection procedure, in order to improve the convergence rate. Other interesting approaches briefly described in (San José, 2005) include those by Yen and Hanzo (Yen, 2001) and Shayesteh et al. (Shayesteh, 2001; Shayesteh, 2003).
In the following sections we propose to use a multiuser detector based on the GA-TS algorithm described in section 3. This detector offers an extremely low computational load as well as a remarkable diversity control based on the on-line adjustment of its internal parameters. This task is achieved by making use of an individuals' fitness dispersion measure based on the Shannon entropy. Based on the ML rule, we develop a GA-TS based multiuser detector that estimates both the channel impulse response (CIR) and the transmitted bit sequences on the basis of the statistics provided by the bank of matched filters at the receiver.

4.2 DS/CDMA system description

Consider a symbol-synchronous binary communication system with K active users, employing normalized modulation waveforms $\left\{ s_i(t) \right\}_{i=1}^{K}$, and signaling through a dispersive channel with AWG noise. User i transmits a sequence of statistically-independent symbols, $b_i(n)$, which modulates the PN sequence, $s_i(n)$, so that the spectrum is spread by a factor N (processing gain) (Glisic, 1997). Thus, the signal transmitted by the ith user is

$$x_i(t) = \sum_{n=0}^{M-1} b_i(n) s_i(t - nT) \tag{11}$$

where T is the signalling interval, M is the number of data bits in a frame transmitted by each user, and the signature waveforms are given by

$$s_i(t) = \sum_{\ell=0}^{N-1} s_{i,\ell} \psi(t - \ell T_c) \tag{12}$$

where $\mathbf{s}_i=(s_{i,0},...,s_{i,N-1})^T$ is the signature vector of user i, $T_c=T/N$ is the chip interval and $\Psi(t)$ represents the energy-normalized chip pulse.
Due to the consideration of synchronous transmission and assuming that users transmit data packets over a single-path frequency-nonselective slowly Rayleigh fading channel, the received baseband signal is given by

$$r(t) = \sum_{i=1}^{K} r_i(t) + g(t) \qquad 0 \le t \le T_f \tag{13}$$

with

$$r_i(t) = \sqrt{E_i}\sum_{n=0}^{M-1} h_i(n)b_i(n)s_i(t-nT) \tag{14}$$

where T_f is the frame duration, $g(t)$ is a zero-mean complex additive white Gaussian noise uncorrelated with $b_i(n)$, $h_i(n)$ denotes the complex CIR coefficient of the ith user and E_i represents the bit energy of user i. We consider, also, the time variation model proposed in (Yen, 2001), where $h_i(n)$ varies over the frame duration according to a specified Doppler frequency, f_d. The unknown variables in Eq. (14) are $b_i(n) \in \{+1,-1\}$ and $h_i(n)$, which denote the nth bit and the corresponding complex CIR coefficient of the ith user, respectively.
At the receiver, a bank of filters matched to the set of users' signature sequences takes samples at every bit interval (see Fig. 6).

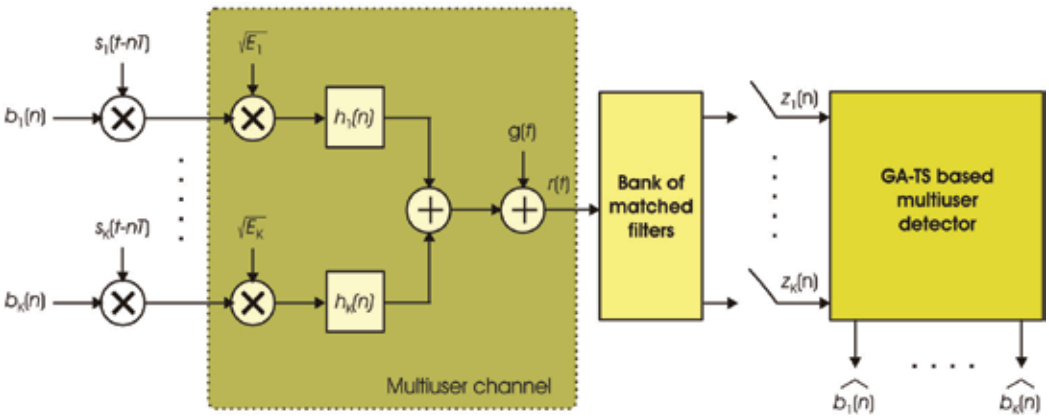


Fig. 6. DS/CDMA multiuser communications system model.

The developed GA-TS-based multiuser detector estimates symbol vector $\boldsymbol{b}(n)=[b_1(n),\ldots,b_K(n)]^T$, containing the symbols transmitted in the nth symbol period. Using the maximization problem as stated in (Yen, 2001), the output of the matched filters can be written as

$$\mathbf{z}(n) = [z_1(n),\ldots,z_K(n)]^T = \mathbf{RH}(n)\mathbf{Eb}(n) + \mathbf{g} \tag{15}$$

where $\mathbf{R}$ is the $K\times K$ user signature sequence cross-correlation matrix, $\mathbf{H}(n) = \mathrm{diag}\left(h_1(n),\ldots,h_K(n)\right)$, $\mathbf{E} = \mathrm{diag}\left(\sqrt{E_1},\ldots,\sqrt{E_K}\right)$, $\mathbf{b}(n) = \left[b_1(n),\ldots,b_K(n)\right]$ and $\mathbf{g} = \left[g_1(n),\ldots,g_K(n)\right]$.

Based on the observation of vector $\mathbf{z}$, it can be shown that the log-likelihood function (LLF conditioned on both the channel matrix $\mathbf{H}(n)$ and the users' data vector $\mathbf{b}(n)$, is given by (Fawer, 1995)

$$\begin{aligned} L(\mathbf{H}(n),\mathbf{b}(n)) = {} & 2\Re\left\{\mathbf{b}(n)^T\mathbf{E}[\mathbf{H}(n)]^*\mathbf{z}(n)\right\} \\ & -\mathbf{b}(n)^T\mathbf{EH}(n)\mathbf{R}[\mathbf{H}(n)]^*\mathbf{Eb}(n) \end{aligned} \tag{16}$$

where symbol "*" denotes the conjugate operation. Thus, the optimal estimates of the diagonal channel gain matrix $\mathbf{H}(n)$ and the vector of transmitted symbols $\mathbf{b}(n)$ are given by

$$(\widehat{\mathbf{H}(n)},\widehat{\mathbf{b}(n)}) = \arg \max_{\mathbf{H}(n),\mathbf{b}(n)} \left\{ L\left(\mathbf{H}(n),\mathbf{b}(n)\right) \right\} \tag{17}$$

subject to a constraint given by the specific model chosen for the time variation of the channel coefficients matrix **H**.
In this work the channel fading is assumed to slow such that coefficients $h_i(n)$ may be considered constant over one signalling interval and the fading is independent for all users. Several models have been used in order to describe the channel characteristics. For example, the Jakes model (Verdú, 1998) or a first-order Gauss-Markov model as given by

$$h_i(n+1) = \alpha h_i(n) + v_i(n) \tag{18}$$

where $\alpha = \exp(2\pi f_d T)$ and $v_i(n)$ is a zero mean white Gaussian variable. Alternatively, this relation between $h_i(n)$ and $h_i(n+1)$ can be expressed as

$$h_i(n+1) = h_i(n) + \Delta_i(n) \tag{19}$$

with $\Delta_i(n)$ being a random variable whose value depends on α and $v_i(n)$. Thus, Eq. (19) constitutes the constraint required for solving the maximization problem given by Eq. (17), and represents the channel model that will be here used.

4.3 Specific fitness calculation and encoding scheme

In the DS/CDMA symbol detection problem the fitness value is computed by substituting the corresponding elements of $\mathbf{u}_i$ into the log-likelihood function L defined in Eq. (16), since both the channel coefficients and the users' transmitted symbols are encoded in $\mathbf{u}_i$:

$$\begin{aligned} \mathbf{u}_i &= [\mathbf{H}_i(n,k) \mid \mathbf{b}_i(n,k)] \\ &= [(h_{i,1}(n,k), h_{i,2}(n,k), \ldots, h_{i,K}(n,k)), (b_{i,1}(n,k), b_{i,2}(n,k), \ldots, b_{i,K}(n,k))] \end{aligned} \tag{20}$$

with parameters i, n, k and K denoting the ith chromosome of the population, the nth signaling interval, the kth generation and the number of active users K, respectively. When the performance of the algorithm is evaluated by means of simulations, the number of users, K, is initially fixed, and it is not allowed to change with respect to the rest of the parameters.
Vector $\mathbf{b}_i(n,k)$ represents a bit string of length K, which is directly binary encoded wit {+1,−1} elements. On the other hand, each complex CIR coefficient in encoded into a binary string of length 22 as depicted in Fig. 7.
Each complex coefficient $h_{i,j}(n,k)$ has both a real and an imaginary part. Each of them, in turn, is represented by a string of 1+10 bits. The first bit (on the left) indicates the sign of the coefficient (with "1" meaning positive sign) and the remaining 10 bits encode the magnitude of the coefficients - with three decimal positions - as shown in the example. Since 10 bits allow numbers between 0 and 1023, a normalization step is used in order to work in the range [0,1].
On the other hand, due to our particular application and encoding scheme, K+1 different positions are randomly generated for possible mutation: one position within the 22 bits of the binary encoding of each channel coefficient $h_{i,j}(n,k)$, $1 \le j \le K$, in Eq. (20), and another one in the part that encodes the users' bits, i.e. $\mathbf{b}_i(n,k)$ in Eq. (20). In each of these K+1 positions, mutation is performed with probability p_m.

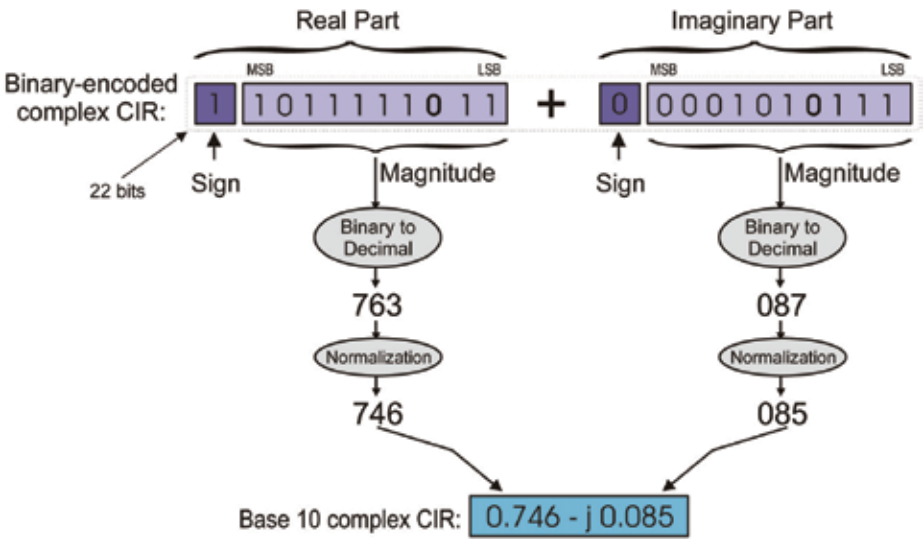


Figure 7. Example of the binary encoding of the channel impulse response coefficients.

Note that, due to the consideration of synchronous transmission and perfect time coordination between the transmitter and the receiver, the detection is achieved by considering separate signal intervals, and, consequently, one GA-TS is executed per interval. As a consequence, the knowledge about the channel impulse response is maintained from one symbol period to another.

4.4 Numerical results

For the simulations, a rectangular chip pulse $\Psi(t)$, AWG noise, Gold-type signatures and a Doppler frequency of 1000 rad/sec are considered. A perfect synchronization is assumed as well. Experiments are performed using binary encoding, K=10, n_p=16, β_m= 0.1 and N=31, if nothing different is specified. Also, binary tournament selection, uniform crossover and an elitism of 3 individuals are used. Memory and simple control of tabu moves are also implemented. The initial probabilities were: p_m=0.1 and p_c=0.05, which proved to be good settings in preliminary experiments.

- **BER performance of the user of interest for different multiuser detection schemes**

Figure 8 shows the BER performance of the user of interest versus $SNR=E_1/N_0$, with a near-far effect of 4 dB (i.e. $E_k/E_1 = 4$ dB, $2 \le k \le K$), for different types of detectors.

First, the proposed GA-TS algorithm is compared to a standard GA in terms of the BER of the user of interest (user 1) versus the signal to noise ratio of this user (E_1/N_0). It can be seen how the GA-TS has a performance close to the limit of the optimal single user detector - a lower bound of the multiuser detector can be found by simulating the BER of the single user receiver in the absence of the other interfering users (Proakis, 1995) - while the standard GA saturates at high SNRs even with higher population sizes and number of iterations – this fact has also been pointed out by other authors, such as Shayesteh in (Shayesteh, 2003).

Simulations show that the GA-TS method needs about 23% of the population needed for the standard GA, and a number of iterations about 15-20%, to get a similar performance, i.e., saving about 70% of time. This value is similar to that reported in (Shayesteh, 2003), where a GA is used to estimate only the vector of users' symbols (in contrast to our approach that jointly estimates both the CIR and the symbols (see section 4.2).

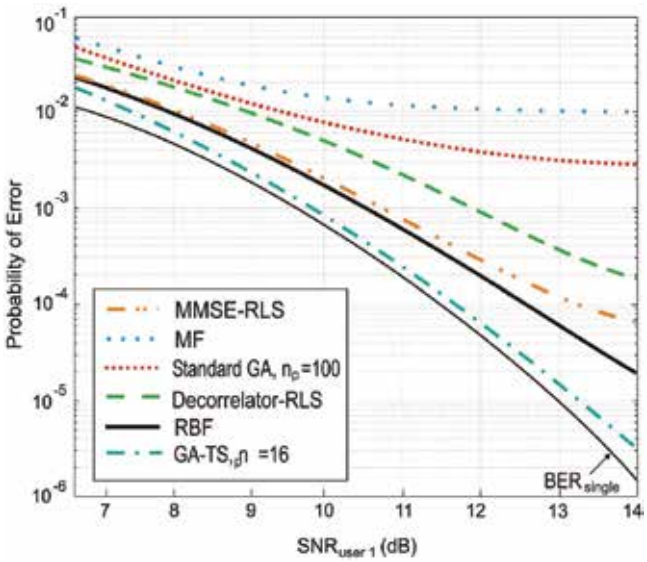


Fig. 8. Probability of error for different types of multiuser detectors. E_k/E_1=4 dB for $k \neq 1$.

The error probabilities of those traditional detectors based on matched filters (MF), the MMSE detector, the decorrelator and an RBF network are also shown for comparison in Fig. 8. The RBF-based MUD chosen for comparison is described in (San José, 2003). As kernel function, the basis function based on the Mahalanobis distance measure is used - its explicit expression can be found in (San José, 2003). The simulations carried out with both the MMSE and the decorrelator detectors, where implemented using a CIR estimation algorithm whose main details and equations can be found in (Cañibano, 2004). For the simulations carried out in this chapter, the RLS version of the equations there obtained was used.

	Standard GA	(Shayesteh, 2003) method	(Yen, 2001) method	Proposed GA-TS
Population size, n_p	60–80	25	40	15
No. generations, n_g	150–250	20	10	20-25
Fitness evaluations	5000	520	400	260

Table 1. Quantitative comparison between various evolutionary multiuser detectors (K=10).

Analyzing the BER plots shown in Figure 8, it can be observed that the proposed GA-TS shows better performance, specially when the SNR is high, than the other compared MUDs. Table 1 compares quantitatively the proposed GA-TS with (Shayesteh, 2003) (for this algorithm the channel is considered to be known), (Yen, 2001) and the standard GA. In case the channel response is known, (Shayesteh, 2003) reports that its algorithm needs about 21% of the population size required in a standard GA (also implemented with known channel response), but using a similar number of iterations. In our simulations, the standard GA used for comparison needs about 150-250 iterations for convergence, while the proposed GA-TS uses only about 20-25. Fig. 8 shows that only a small degradation is observed with respect to the single-user bound when the proposed GA-TS multiuser detector is used.

- **Population size vs number of active users**

Analyzing how the number of active users affects the proposed detector, we find that the BER performance gradually degrades upon increasing the number of users, due to the limited population size n_p, which becomes too small for adequately exploring a larger space. Furthermore, in comparison with the "brute-force" optimum ML MUD requiring $2^K=2^{10}=1024$ fitness function evaluations, the proposed detector is substantially less complex, requiring only $n_p \, x \, n_g$ = 15 x 20 (or 23) = 300 (or 375) evaluations, yet performing close to the optimum ML MUD. This fact clearly manifests that the computational complexity of the GA-TS algorithm does not depend exponentially on K. In fact, with 20 users, a population size of 125 - keeping n_g constant - allows the detector to attain a near-optimal performance. This means an increase by a factor of 150/15=8.33 in the computational load. Furthermore, in contrast to the reduced tree-search type algorithms, our detector does not require any memory capacity, since all the information related to previous generations can be erased.

It should be noticed that this increment in the number of active users could also be addressed by increasing the number of iterations n_g while keeping constant the size of the population n_p. These two solutions could be considered also if the SNR decreases. This interplay between n_g and n_p has also been pointed out in (Shayesteh, 2003). However, it is difficult to find a general rule that could easily quantify this equivalence between n_p and n_g since it greatly depends on factors like the number of users K, the ratio E_k/N_0, etc. In the aforementioned example, with (K=20, n_p=125, n_g=20) another configuration of the GA-TS with (K=20, n_p=75, n_g=40) obtains similar results.

Next, the BER performance for K=15 users is analyzed. Because of the higher number of users to be optimized, we increased the population size. With n_p = 35, a significant degradation was observed. This situation was due to the limited population size, which was too small for optimizing the number of involved variables. However, when increasing n_p up to 45, the performance became near optimum, though the increase in complexity is really low in comparison with the computational load of the conventional optimum MUD, whose complexity will now be increased by a factor of 32.

No. of users (K)	Population size (n_p)	No. of generations (n_g)
10	15 / **40**	20 / **10**
15	45 / **75**	20 / **10**
20	125 / **160**	20 / **10**

Table 2: Population size (n_p) and number of generations (n_g) required for different number of active system users (K) for the proposed GA-TS algorithm. Bold values correspond to the (Yen, 2001) method.

These results are summarized in Table 2. Values in bold typesetting correspond to the GA-based detector developed by Yen and Hanzo (Yen, 2001). It can be seen that the GA-TS scheme requires a smaller population size at the expense of a higher number of iterations.

- **BER vs number of active users**

Now, in Figure 9, the BER for different receiver structures (GA-TS, RBF, MF and MMSE) as a function of the number of users is compared. For this simulation, a perfect power control has been considered: $E_i = E_j$, $1 \le i,j \le K$, $i \ne j$.

The MMSE structure and the linear MF detector are outperformed by the nonlinear RBF and GA-TS receivers. The MF suffers from MAI and the MMSE lacks stability to perform a

nonlinear decision boundary. The choice of the Mahalanobis distance in the RBF basis function allows some advantages over the Euclidean one. The Euclidean-RBF performance tends towards the MMSE receiver performance due to the non-spherical shape of the multidimensional clusters.

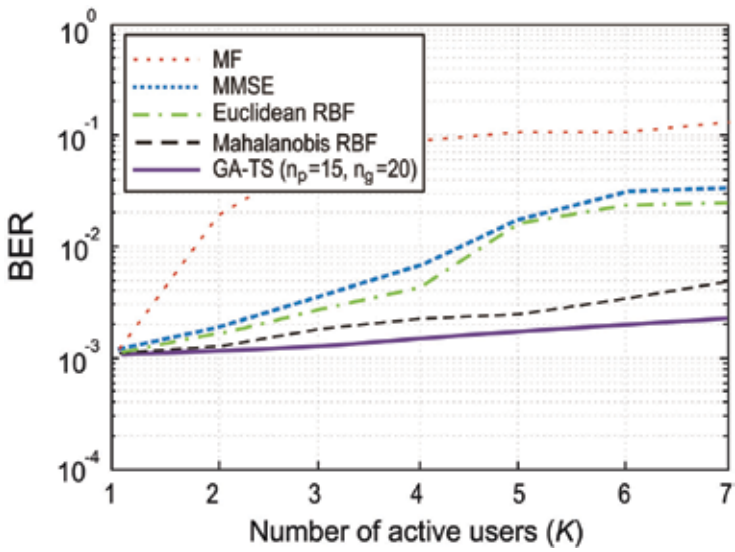


Fig. 9. Probability of error vs number of active users.

Figure 9 shows that the proposed GA-TS method and the Mahalanobis RBF detector have similar performance. However, it must be pointed out that the complexity of the RBF detector used for comparison is much higher. This RBF-based detector - described in (San José, 2003) - needs a training period if the channel response is unknown and its complexity (number of nodes) increases exponentially with the number of active users.

Quantitatively, the computational requirements (in terms of floating point operations in MATLAB) have been experimentally shown to be at the same level when just a few users are present ($K \leq 3$). However, for higher values of K, the proposed GA-TS receiver clearly outperforms the RBF-based detector in terms of computational complexity.

- **Joint channel estimation and symbol detection performance**

In this section we compare the performance of the proposed GA-TS-based multiuser detector when (i) the channel impulse response is estimated using the GA-TS method (joint estimation of the users' symbols and the CIR) and (ii) the CIR is perfectly known, i.e. the true channel coefficients are substituted in Eq. (16).

Figure 10 shows the different BERs obtained with and without estimation of the channel impulse response coefficients. The single-user bound shown is computed as $P_e = 0.5(1-(\gamma/(1-\gamma))^{0.5})$, where $\gamma = E_k/N_0$ (valid for BPSK).

Two comments apply to this figure: (i) the higher n_g and n_p are chosen, the closer are the BER curves for both cases (with and without CIR estimation), and (ii) both BERs are similar up to a threshold value of the E_k/N_0 parameter that depends on the election of n_p and n_g. For instance, in the case with n_p=35 and n_g=15, BERs are very similar whenever $E_k/N_0 \leq 26$ dB. In the other case, with n_p=25 and n_g=10, this threshold is located around 14 dB.

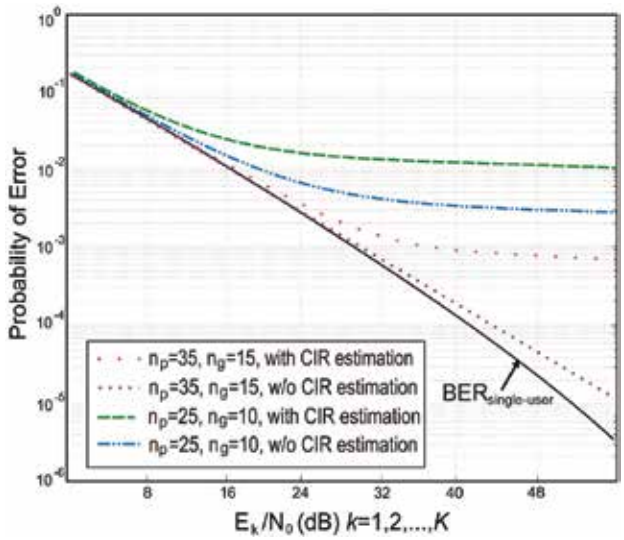


Fig. 10. BER performance comparison between the GA-TS algorithm in which both the channel impulse response and the users' symbols are jointly estimated, and that in which only the users' symbols are unknown. (CIR: channel impulse response).

- **Near-far resistance**

Next, Fig. 11 characterizes the near-far resistance of the proposed detector in terms of the BER of the desired user. Simulations used n_p=30 and n_g=15. The user of interest is user 1 and the averaged received bit energies of all the other users were set to 0, 5, 10 and 15 dB higher. Simulations of the decorrelator detector were performed with RLS parameter estimation, while the GA-TS detector was implemented with the joint estimation of the channel and the symbols. It can be seen that for $E_k/E_1 \leq 10$ dB, the proposed detector is near-far resistant up to $E_1/N_0 \approx 15$ dB. Beyond that, a significant BER performance degradation is observed.

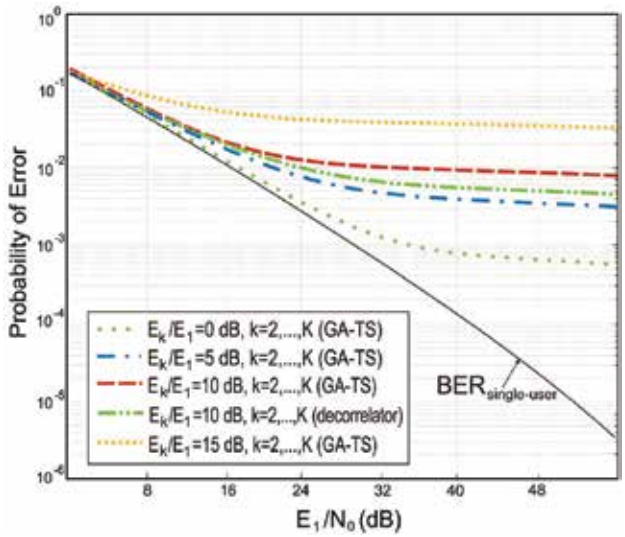


Fig. 11. BER performance for K=10 users with E_k/E_1=0, 5, 10 and 15 dB for k=2,…,K. User 1: user of interest.

For the sake of comparison, the BER performance corresponding to the decorrelator detector implemented using a RLS-type estimation algorithm, is also plotted for the case with E_k/E_1=10 dB. In this case, the decorrelator detector shows a little better performance at the expense of a much higher complexity.

- **Entropy-guided parameter tuning**

Finally, some experiments were carried out in order to study the influence of the entropy-based tuning up process of the GA-TS parameters. Figure 12 represents the evolution of the mean population fitness (mean value after 100 runs) of: (i) the individuals of the elite, (ii) all those individuals not belonging to the elite. For both cases, the evolution is drawn with and without entropy-guided adjustment for the first 50 iterations of the algorithm. For both sets of individuals, the mean fitness approaches the maximum before when the entropy is monitored and the probabilities p_c and p_m are dynamically adjusted. Furthermore, redundant individuals are more efficiently eliminated when the fitness entropy is used to adjust the genetic operators. This way, the diversity of the population increases, and, simultaneously, the probability of getting trapped into suboptimal solutions is reduced. This capability also affects the population size required to get a specific BER, since: (i) very similar individuals do not represent any advantage at the time of searching the solutions space, and (ii) almost no improvement (in terms of mean population fitness) occurs during the exploration phases. Thus, many fitness evaluations can be saved during these periods. Making use of this fact, the number of evaluations can be lowered around 25%.

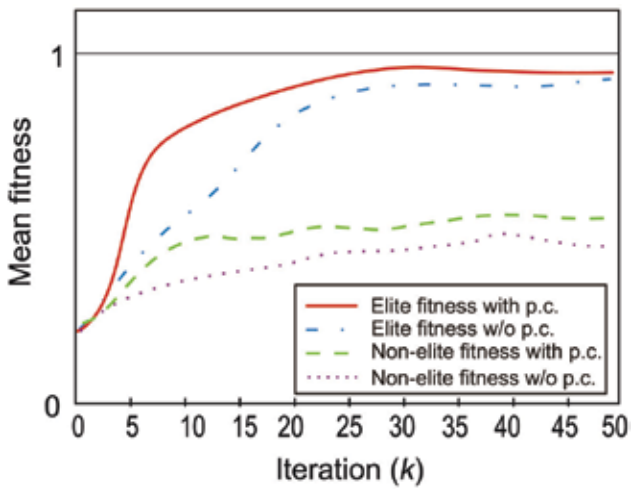


Fig. 12. Evolution of the population fitness with and without parameter control (p.c.).

4.5 Conclusions

A novel hybrid genetic-algorithm / tabu-search (GA-TS) algorithm with an extremely low complexity has been proposed for DS/CDMA multiuser detection. It requires no training period, applies the genetic operators only very few times and saves a substantial amount of fitness evaluations, which is a critical factor in real-world applications. The performance is close to the limit of the single user detector and its complexity remains much smaller than that of traditional approaches and those based on standard GAs. The entropy-guided

procedure to on-line adjust the probabilities of the genetic operators controls the diversity of the population; this leads to a reduction of the population size required to attain a certain performance and a high probability of achieving a reliable convergence result.

5. Application II: tuning-up of a DT-MRI tracking algorithm

5.1 Motivation and problem description

The Diffusion Tensor Magnetic Resonance Imaging (DT-MRI) technique measures the diffusion of hydrogen atoms within water molecules in 3D space. Since in cerebral white matter most random motion of water molecules are restricted by axonal membranes and myelin sheets, diffusion anisotropy allows depiction of directional anisotropy within neural fiber structures (Ehricke, 2006). The DT-MRI technique has raised great interest in the neuroscience community for a better understanding of the fiber tract anatomy of the human brain. Among the many applications that arise from tractography we find: brain surgery (knowing the extension of the fiber bundles could minimize the functional damage to the patient), white matter visualization using fiber traces (for a better understanding of brain anatomy) and inference of connectivity between different parts of the brain (useful for functional and morphological research of the brain).

Most of DT-MRI visualization techniques focuses on the integration of sample points along fiber trajectories (Mori, 2002), using only the principal eigenvector of the diffusion ellipsoid as an estimate of the predominant direction of water diffusion (Ehricke, 2006). However, these methods may depict some fiber tracts which do not exist in reality or miss to visualize important connectivity features, e.g. crossing or branching structures. In order to avoid misinterpretations, the viewer must be provided with some information on the uncertainty of every depicted fiber and of its presence in a certain location. In (San José, 2007) we proposed an estimation algorithm that takes into account the whole information provided by the diffusion matrix, i.e., it does not only consider the principal eigenvector direction but the complete 3D information about the certainty of continuing the path through every possible future direction. An improved version of this algorithm was developed in (San José, 2007b). This article included two main aspects: (i) a procedure for on-line adapting the number of *offspring paths* emerging from the actual voxel, to the degree of anisotropy observed in its proximity (this strategy was proved to enhance the estimation robustness in areas where multiple fibers cross while keeping complexity to a moderate level), and (ii) an initial version of a neural network (NN) for adjusting the parameters of the algorithm in a user-directed training stage. Subsequent work (San José, 2008) studied with more detailed the architecture of the neural network and numerically evaluated its tracking capability, robustness and computational load when used with both synthetic and real DT-MR images. This work showed that in many cases, such as real images with low SNR, a huge computational load was required.

Now we propose to use an evolutionary computation-based approach - the GA-TS algorithm - for tuning-up the parameters of the tracking algorithm instead of using the neural network. The main aim is to adjust the parameters with a less complex procedure and to obtain a robust and efficient tracking algorithm. The required human intervention time can also be reduced. Numerical results will prove that the GA-TS approach leads to similar and even better convergence results while offering much lower computational requirements.

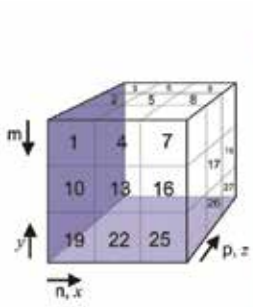
5.2 Brief tracking algorithm description

Since our main purpose is the development of an algorithm to adjust the parameters of a tracking algorithm, it is necessary to outline the head expressions of this algorithm - for the sake of brevity, the reader interested in a detailed development of this tracking algorithm can see the corresponding sections in (San José, 2007). Consequently, this section presents a brief summary of the method. The algorithm uses probabilistic criteria and iterates over several points in the analyzed volume (the points given by the highest probabilities in the previous iteration). The process starts in a user-selected seed voxel, V_0.

Every iteration, the method evaluates a set of parameters related to the central voxel of a cubic structure similar to that shown in Figure 13, left. The central point, V_c, (No. 14 in the figure) represents the last point of the tract being analyzed. In the first iteration, $V_c = V_0$.

- **Basic concepts**

First, a measure P_i, $i \in \{valid\ points\}$, is evaluated based on the probability of going from voxel V_c to voxel V_i. This probability takes into account the eigenvalues and eigenvectors available at point V_c from the DT-MR image diffusion matrix. In order to calculate this probability, the information shown in Fig. 13, right, is used.



mnp	$\pi_m\pi_n\pi_p$	mnp	$\pi_m\pi_n\pi_p$	mnp	$\pi_m\pi_n\pi_p$
1°) 111	(---)	10°) 211	(0--)	19°) 311	(+--)
2°) 112	(--0)	11°) 212	(0-0)	20°) 312	(+-0)
3°) 113	(--+)	12°) 213	(0-+)	21°) 313	(+-+)
4°) 121	(-0-)	13°) 221	(00-)	22°) 321	(+0-)
5°) 122	(-00)	14°) 222	(000)	23°) 322	(+00)
6°) 123	(-0+)	15°) 223	(00+)	24°) 323	(+0+)
7°) 131	(-+-)	16°) 231	(0+-)	25°) 331	(++-)
8°) 132	(-+0)	17°) 232	(0+0)	26°) 332	(++0)
9°) 133	(-++)	18°) 233	(0++)	27°) 333	(+++)

Fig. 13. Modifications of indices (m,n,p) when moving from V_c to the neighbouring voxel V_i, $1 \le i \le 27$, $i \ne 14$.

The table shows, for every voxel shown in Fig. 13, left, the changes that must occur in indices (m,n,p), when a tract goes from voxel V_c to voxel V_i. For instance: when going from point No. 14 to point No. 17, coordinates m and n increase by 1, and p remains the same. This is represented in the table with "π_m,π_n,π_p = (++0)". With this information, the probability of each possible destination V_i can be calculated taking into account the projection of each of the eigenvectors to each of the directions defined in the triplet π_m, π_n, π_p. Besides, each projection is weighted by the corresponding eigenvalue λ. Thus, in the previous example, P_i should be calculated as $P_i = V_{1y}\,\lambda_1 + V_{2y}\,\lambda_2 + V_{3y}\,\lambda_3 + V_{1z}\,\lambda_1 + V_{2z}\,\lambda_2 + V_{3z}\,\lambda_3$, where $V_{j\alpha}$ represents the α-component of eigenvector j, $1 \le j \le 3$, $\alpha \in \{x,y,z\}$.

The axes reference criterion for the (x,y,z) vector components is also shown in Fig. 13. Note that, for this calculus, the sign "-" in the triplet is equivalent to sign "+". In order to properly calculate P_i, it must be weighed by 0.33 if there are no zeros in triplet i, and by 0.5 if there is one zero.

- **Anisotropy and local probability**

The following anisotropy index is used in the algorithm:

$$fa = \sqrt{\frac{(\lambda_1 - \lambda_2)^2 + (\lambda_2 - \lambda_3)^2 + (\lambda_1 - \lambda_3)^2}{2(\lambda_1^2 + \lambda_2^2 + \lambda_3^2)}}, \tag{21}$$

where $\lambda_1 \geq \lambda_2 \geq \lambda_3$. When both $fa(V_c)$ and $fa(V_i)$ do not exceed a certain threshold, then point V_i is eliminated as a possible destination point.
Taking into account both P_i and the anisotropy given by Eq. (21), the local probability of voxel i is defined as

$$P_i^{'} = a \cdot \mu_1 \cdot fa(V_i) + (1-a) \cdot \mu_2 \cdot P_i, \qquad 0 < a < 1 \tag{22}$$

where parameter a allows the user to give a higher relative weight to either the anisotropy or the local probability, and μ_1 and μ_2 are scaling factors (normally, 1 and 1000, respectively). The set of values $P_i^{'}$ is properly normalized so that they can be interpreted as probabilities.

- **Eigenvectors and direction considerations**

Besides these considerations, the final probability of voxel i makes also use of the so-called *smoothness parameters* - described in (Kang, 2005) - which judge the coherence of fiber directions among the trajectories passing through voxel V_c. The mathematical expressions of these four parameters, $\{sp_i\}_{i=1}^{4}$, as well as their geometrical meaning, is explained in (San José, 2008). They measure the angles between the directions that join successive path points, as well as the angles between these directions and the eigenvectors associated to the largest eigenvalues found in those voxels. sp_2, sp_3 and sp_4 are used to maintain the local directional coherence of the estimated tract and avoid the trajectory to follow unlikely pathways. The threshold for sp_1 is set such that the tracking direction could be moved forward consistently and smoothly, preventing the computed path from sharp transitions.
Next, the following parameter is calculated for every valid point whose smoothness parameters satisfy the four corresponding threshold conditions,

$$P_i^{''} = b(\xi_1 sp_1 + \xi_2 sp_2 + \xi_3 sp_3 + \xi_4 sp_4) + (1-b)P_i^{'} \tag{23}$$

where, ξ_1, ξ_2, ξ_3 and ξ_4 are the corresponding weights of the smoothness parameters (normally, 0.25), and b stands for a weighting factor.

- **Path probabilities**

Probabilities $P_i^{''}$ can be recursively accumulated, yielding the probability of the path generated by the successive values of V_c,

$$P_i^{'''} = P_i^{''} / \sum_i P_i^{''} \tag{24}$$

with k being the iteration number, and $P_i^{''} = P_i^{'} / \sum_i P_i^{'}$. At the end of the visualization stage, every estimated path is plotted with a colour depending on its probability P_p.

- **Final criterion and pool of "future seeds"**

A pool of voxels is formed by selecting, at the end of each iteration, the s best voxels according to Eq. (23). The first voxel of the pool becomes the central voxel V_c at next iteration, expanding, this way, the current pathway.
As proposed in (San José, 2008), the value of s is adjusted depending on the degree of anisotropy found in current voxel V_c and its surroundings. When this anisotropy is high, it

means that a high directivity exists in that zone, and the probability that V_c belongs to a region where fibers cross is really low. Consequently, s takes a small value (1, 2 or 3). On the other hand, if V_c is found to be situated in a region of high anisotropy, the probabilities of having fibers crossing or branching is higher. In this case, it is interesting to explore various paths starting in V_c. This can be achieved by setting parameter s to a higher value.

- **Parameters to be estimated using the GA-TS algorithm**

We propose to use the previously developed GA-TS procedure for adjusting the parameters of the algorithm $\Omega = (a, b, \mu_1, \mu_2, \xi_1, \xi_2, \xi_3, \xi_4)$, instead of using the complex and time consuming neural network proposed in (San José, 2007b; San José, 2008). This adjustment is useful when the algorithm is applied to a different part of the brain (fiber bundles) or even to the same portion but having been scanned with under different conditions. In these cases, the volume of interest will have a different smoothness and anisotropy characterization.

- **Estimation procedure**

The user-aided estimation procedure works as follows: first, the user is requested to manually draw a sample fiber path as well as to compare this fiber path to those estimated by the GA-TS method during its first stages. Specifically, the steps for the estimation of the parameters are: (i) the user manually draws a sample fiber path, $\mathbf{r}_u$, (ii) the GA-TS scheme starts with a randomly generated population of n_p=10 individuals $\left\{\mathbf{u}_i\right\}_{i=1}^{n_p}$, each of them being a possible binary representation of parameters Ω, (iii) the tracking algorithm of section 5.2 is applied n_p times, each of them with the set of parameters represented by each GA-TS's individual. This way, n_p different paths $\mathbf{r}_i$ are obtained, (iv) every path $\mathbf{r}_i$ is compared with $\mathbf{r}_u$ and given a fitness value λ_i, (v) iterate the GA-TS algorithm during n_g=25 generations and then go to step (ii).

Every time that the fiber paths are obtained at step (ii) the user must compare them to his sample $\mathbf{r}_u$ and, in case he finds that a tract $\mathbf{r}_j$, $1 \le j \le n_p$, is better than his first estimation $\mathbf{r}_u$, then $\mathbf{r}_j$ becomes the new reference path $\mathbf{r}_u$. At the end, solution Ω is obtained from the encoding of the fittest individual.

Though this scheme seems initially complicated, experiments show that a few iterations lead to sets Ω that allow to obtain good results when used in the tracking algorithm. The user will not have to assign too many fitness values or to perform many comparisons. The extremely reduced size of the population and the low number of generation per GA-TS execution, derive in moderately short training periods.

5.3 Numerical results

In order to evaluate the proposed algorithm (parameter tuning + tracking), both synthetic and real DT-MR images have been used. For the sake of comparison, we have used the same test images as in (San José, 2008).

- **Synthetic images**

Figure 14 shows four different synthetic DT-MRI data defined in a 50×50×50 grid (we will refer to them as "cross", "earth", "log" and "star"). To make the simulated field more realistic, Rician noise (Gudbjartsson, 1995) was added in the diffusion weighted images which were calculated from the Stejskal-Tanner equation using the gradient sequence in (Westin, 2002) and a b-value of 1000. The desired noisy synthetic diffusion tensor data was obtained using an analytic solution to the Stejskal-Tanner equation.

Satisfactory tracing results for the first three cases can be found in (San José, 2007), where a simpler algorithm was used. For the sake of brevity, in this paper we have worked with the

most complex case, the *star*. This image consists of six orthogonal sine half-waves, each of them with an arbitrary radius. Under this scenario the diffusion field experiments variations with the three coordinate axes and there exists a crossing region.

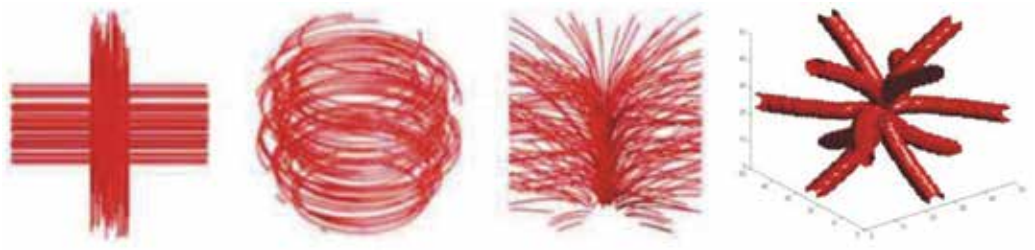


Fig. 14. Synthetic DT-MR images used for testing the proposed algorithm: "cross" (left), "earth", "log" and "star" (right).

Three different tracking results are shown in Fig. 15, each of them for a different seed V_0={S_1,S_2,S_3}. Blue tracts were obtained with an algorithm were parameters were estimated with a NN (San José, 2008), while green ones correspond to the estimation using the proposed GA-TS algorithm.

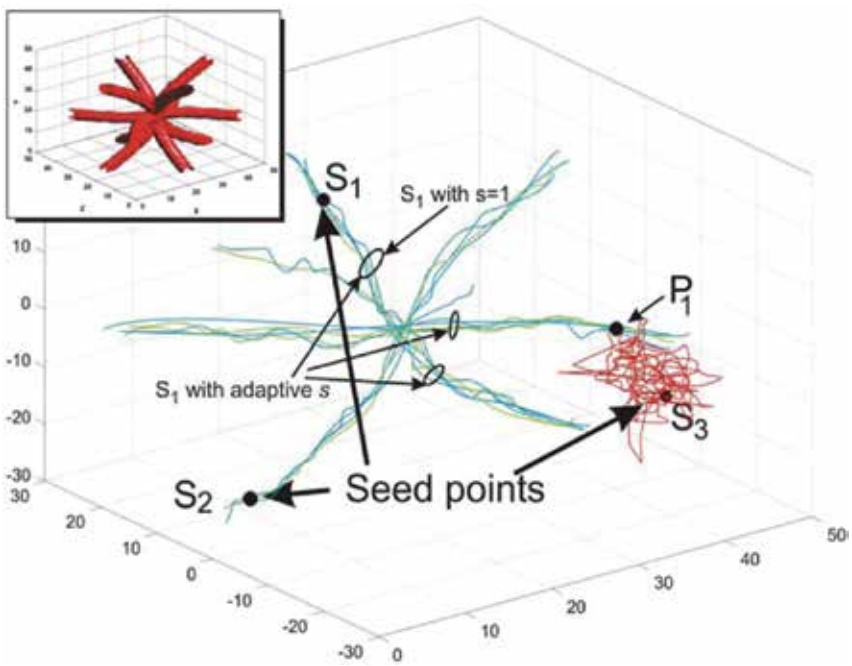


Fig. 15. Tracking results for the "star" synthetic DT-MR image. Black: seed points. Blue: fiber paths obtained using adjustment with NN, Green: paths using estimation with the proposed GA-TS. Red: extrinsic voxels. Initial seeds V_0={S_1,S_2,S_3}. Top left: original synthetic image.

It can be seen that in both cases the path estimates pass through isotropic zones where different fiber bundles cross. It is also appreciated how both methods differentiate between the totally isotropic zones extrinsic to the tracts and the fiber bundles.

The differentiation between voxels belonging to a fiber or to a very isotropic area, respectively, is attained by mapping the path probabilities given by Eq. (24) into a colour scale and classifying them according to some fixed thresholds. Notice that seeds S_1 and S_2 belong to the intrinsic volume (voxels with a very high anisotropy). In this case, both methods move through the most probable direction following the main direction of the star in each situation. When extrinsic point S_3 is selected as seed, the algorithms explore in the neighbouring voxels until they find a voxel with a high anisotropy value (point P_1). Once P_1 is found, the tracking algorithm proceeds as in the case of S_1 and S_2. Fig. 15 shows how the algorithm finds the proper fiber path whatever (extrinsic or intrinsic) seed voxel is chosen, for both methods of parameters' estimation.

			SNR (dB)					
			5	10	15	20	25	30
Image	Cross	NN	78.3/**82.8**	89.7/**93.6**	92.1/**94.3**	98.3/**98.7**	99.0/**99.0**	100/**100**
		AG	81.2/**85.7**	93.6/**94.5**	94.9/**96.1**	98.8/**98.7**	99.0/**100**	100/**100**
		[3]	76.8	89.0	90.7	97.0	100	100
	Earth	NN	77.7/**76.2**	88.6/**87.5**	89.9/**89.0**	98.2/**98.2**	99.0/**99.0**	100/**100**
		AG	82.1/**79.6**	89.6/**92.5**	93.5/**94.0**	98.8/**98.9**	99.0/**100**	100/**100**
		[3]	74.4	83.2	85.0	97.3	99.2	100
	Log	NN	71.0/**69.7**	82.1/**81.0**	86.1/**85.5**	96.0/**95.8**	98.0/**97.8**	100/**100**
		AG	75.0/**75.2**	85.2/**85.0**	89.9/**87.7**	97.0/**97.2**	98.2/**98.4**	100/**100**
		[3]	68.8	78.3	85.2	96.0	98.0	100

Table 3. Convergence performance for different SNRs values. Cell values represent percentage of right convergence for two configurations of the algorithm: s=1/$\boldsymbol{s}$ = **4**. Each cell shows: top: NN-estimation, middle: GA-TS estimation, bottom: Bayesian tracking (Friman, 2005).

Next, the robustness of the tracking algorithm with both parameter estimation methods is now studied. For the sake of brevity, these experiments were run with parameter s kept constant during the fiber tract estimation (see "*Final criterion and pool of future seeds*" in section 5.2).

The convergence performance for different SNRs is shown in Table 3. The first row in each cell corresponds to tracking results when parameters where estimated using the NN, the second contains the results when the proposed GA-TS is used for this estimation, and the third one shows the values obtained with a slightly modified version of the Bayesian method proposed in (Friman, 2005).

It can be seen that both algorithms (with NN- and GA-TS-based adjustment) converge properly within a wide range of SNRs, with the GA-TS version showing a convergence gain of about 3-6% in all cases. The percentage values for the "cross" and the "earth" test images are very close, while for the "log" case both algorithms exhibit a slightly lower convergence. Comparing our methods with the Bayesian approach, we see that the proposed tracking algorithm performs slightly better when the SNR is low, while the three methods tend to similar results with high SNRs.

Analyzing the simulations of the synthetic images considered, it is seen that convergence results improve whenever the MR image contains branching or crossing areas - as it is the case in real DT MR images. This is the case of our "cross" image. For this image, the convergence results are improved ~ 5% when parameter s is modified according to the anisotropy. Besides, for these studied cases, we see that the influence of the procedure that adapts s is higher for low SNRs.

- **Real images**

The proposed tracking algorithm has also been applied to real DT-MR images. Specifically, we have selected the *corpus callosum* of the brain (see Fig. 16).
Simulation results show that whichever parameters tuning-up method is used, the algorithm is able to follow the main fiber bundle directions without getting out of the area of interest. Fig. 16 shows some bundles of properly estimated tracts. Red/green color indicates high/low certainty.

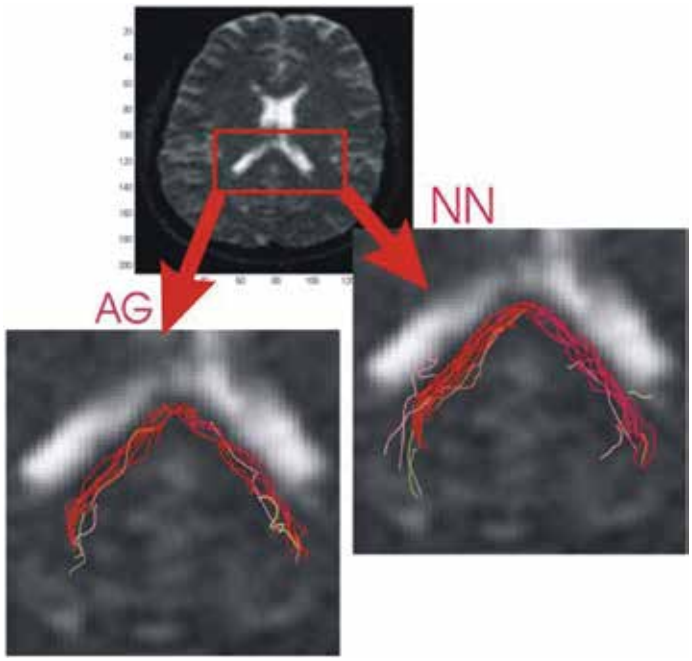


Fig. 16. Tracking results for the *corpus callosum* area of the human brain. Left: tracts obtained with the tracking algorithm tuned-up with the proposed GA-TS method, Right: parameter estimation with neural network.

- **Final remarks**

The proposed parameter estimation procedure is useful when the volume being varies. For instance, with just 5-10 training iterations (repetitions of the "training procedure"), in synthetic images, or 8-16, in real images, the parameters of the algorithm are fine-tuned so as to get satisfactory results. Note that these previous training times are: (i) always inferior to those required by the NN-based method proposed in (San José, 2007b; San José 2008), (ii) always much inferior to the time required to heuristically adjust the parameters, (iii) only required when the scanning conditions vary.

5.4 Conclusions

Numerical simulations have shown that the tracking algorithm that has been tuned-up using the proposed GA-TS-based method is capable of estimating fiber tracts both in synthetic and real images. The robustness and convergence have been studied for different image qualities (SNRs). Results show a convergence gain of about 3-6% with respect to our previous work (San José, 2007b; San José, 2008).

The experiments carried out show that an efficient parameter adjustment in conjunction with precise rules to manage and update both the pool of future seeds and the memory with the tabu list, lead to: (i) a better use of computational resources, (ii) a better performance in regions with crossing or branching fibers, and (iii) a minimization of the required human intervention time. The method has been tested with synthetic and real DT-MR images with satisfactory results, showing better computational and convergence properties even than already existing Bayesian methods such as (Friman, 2005).

6.Acknowledgements

The author would like the Spanish Education Ministry for Grant TEC2007-67073/TCM, that partially funded this work.

7. References

Bhandari, D., Murthy, C.A, Pal, S.K. (1996). Genetic algorithm with elitist model and its convergence, *International Journal on Pattern Recognition and Artificial Intelligence*, Vol. 10, No. 6 (1996) 731–747.

Cañibano, N. & San José-Revuelta, L.M. (2004). Analysis of a Bayesian multiuser detector for non-data-aided CDMA communications, *Proc. IEEE Workshop on Machine Learning and Signal Processing, MLSP'04*, 2004, Sao Lluis, Brazil.

Ergün, C., Hacioglu, K. (2000), Multiuser detection using a GA in CDMA communications systems, *IEEE Trans. on Comm.*, Vol. 48, No. 8 (2000) 1374–1383. ISSN:

Ehricke, H.H., Klose, U., Grodd, U. (2006). Visualizing MR diffusion tensor fields by dynamic fiber tracking and uncertainty mapping, *Computers & Graphics*, Vol. 30 (2006) 255–264.

Faigle, U. & Kern, W. (1992). Some Convergence Results, *Journal on Computing*, Vol. 4 (1992) 32-37.

Fawer, U., Aazhang, B. (1995). A multiuser receiver for CDMA communications over multipath channels, *IEEE Trans. on Comm.*, Vol. 43, (1995) 1556–1565.

Fox, B.L. (1993). Integrating and accelerating tabu search, simulated annealing and genetic algorithms, *Annals of Operations Research*, Vol. 41 (1993) 47-67.

Friman, O. & Westin, C.-F. (2005). Uncertainty in white matter fiber tractography, *Proceedings of the MICCAI 2005, LNCS 3749*, 107-114, 2005.

Ghosh, S.C., Sinha, B.P., Das, N. (2003). Channel assignment using genetic algorithm based on geometric symmetry, *IEEE Transactions on Vehicular Technology*, Vol. 52, No. 4 (2003) 860–875.

Glisic, S., Vucetic, B. (1997). *Spread Spectrum CDMA Systems for Wireless Communications*, Artech House Publishers, ISBN, UK.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, Vol. 13 (1986) pp. 533-549.

Glover, F. (1989). Tabu Search, Part I, *ORSA Journal on Computing*, Vol. 1 (1989) 190-206.

Glover, F. (1990). Tabu Search, Part II, *ORSA Journal on Computing*, Vol. 2 (1990) 4-32.

Glover, F. (1992) Private communication.

Gudbjartsson, H. & Patz, S. (1995). The Rician distribution of noisy MRI data, *Magnetic Resonance in Medicine*, Vol. 34 (1995) 910–914.

Hansen, P. (1986). The steepest ascent mildest descent heuristic for combinatorial programming, *Proc. Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy, 1986.

Hertz, A. (1995). A tutorial on tabu search, *Proc. of Giornate di Lavoro AIRO'95 (Enterprise Systems: Management of Technological and Organizational Changes)*, pp. 13-24, Italy, 1995.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press (Ann Arbor).

Juntti, M.J., Schlosser, T., Lilleberg, J.O. (1997). GAs for multiuser detection in synchronous CDMA, *Proc. IEEE Int'l Symp. on Inf. Theory*, p. 492, 1997.

Kang, N., Zhang, J., Carlson, E.S., Gembris, D. (2005), White matter fiber tractography via anisotropic diffusion simulation in the human brain, *IEEE Transactions on Medical Imaging*, Vol. 24 (2005) 1127–1137.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*, The MIT Press, Cambridge, MA.

Mori, S. & van Zijl, P.C.M. (2002). Fiber tracking: principles and strategies – A technical review, *Nuclear Magnetic Resonance in Biomedicine*, Vol. 15 (2002) 468–480.

Oppacher F. & Wineberg, M. (1999). The Shifting Balance Genetic Algorithm: Improving the GA in dynamic environment, *Proc. Genetic and Evol. Comput. Conf.*, W. Banzhaf et al. (Eds.) Vol. 1, pp. 504–510.

Proakis, J.G. (1995). *Digital Communications*, McGrawHill Int'l Editions, New York.

San José-Revuelta, L.M. & Cid-Sueiro, J. (2003). Bayesian and RBF structures for wireless communications detection, *Proc. IEEE Workshop on Neural Networks and Signal Processing, NNSP'03*, pp. 749–758, Toulouse, France.

San José-Revuelta, L.M. (2005). Entropy-guided micro-genetic algorithm for multiuser detection in CDMA communications, *Signal Processing*, Vol. 85, No. 8, (August 2005) 1572-1587, ISSN 0165-1684.

San-José-Revuelta, L.M., Martín-Fernández, M., Alberola-López, C. (2007). A new proposal for 3D fiber tracking in synthetic diffusion tensor magnetic resonance images, *Proceedings of the IEEE Int'l Symp. on Signal Processing and its Applications, ISSPA'07*, Sharjah, United Arab Emirates, 2007.

San-José-Revuelta, L.M., Martín-Fernández, M., Alberola-López, C. (2007b). Neural-network assisted fiber tracking of synthetic and white matter DT-MR images, *Proceedings of the World Congress on Engineering 2007, WEC 2007*, pp. 618-623, London, United Kingdom, July 2007.

San-José-Revuelta, L.M., Martín-Fernández, M., Alberola-López, C. (2008). Efficient tracking of MR tensor fields using a multilayer neural network, *IAENG International Journal of Computer Science*, Vol. 35, No. 1 (2008) 129–139.

Shayesteh, M.G., Menhaj, M.B., Nobary, B.G. (2001). A new modified GA for multiuser detection in DS/CDMA sytems, *Proc. Comput. Intell., Theory and Appl., 7th Fuzzy Days*, 608-618, Dortmund, Germany, 2001.

Shayesteh, M.G., Menhaj, M.B., Nobary, B.G. (2003). A modified Genetic Algorithm for multiuser detection in DS/CDMA systems, *IEICE Trans. on Comm.*, Vol. E86-B, No. 8 (2003) 2377–2388.

Shimodaira, H. (1999). A Diversity Control Oriented Genetic Algorithm (DCGA): Development and experimental results, *Proc. Genetic and Evol. Comput. Conf.*, W. Banzhaf et al. (Eds.) Vol. 1 (1999) 603–611.

Tsutsui, S., Fujimoto, Y., Ghosh, A. (1997). Forking Genetic Algorithms: GAs with search space division schemes, *Evolutionary Computation*, Vol. 5 (1997) 61–80.

Ursem, R.K. (2002). Diversity-guided Evolutionary Algorithms, *Proc. Int'l Conf. on Parallel Problem Solving from Nature VII, PPSN VII*, pp. 462–471, Granada, Spain, 2002.

Verdú, S. (1998). *Multiuser Detection*, Cambridge University Press, Cambridge, UK.

Wang, X.F., Lu, W.S., Antoniou, A. (1998). A Genetic Algorithm-based multiuser detector for multiple access communications, *Proc. IEEE Int'l Symp. on Circuits and Systems, ISCAS'98*, pp. 534-537, 1998.

de Werra, D. & Hertz, A. (1989). Tabu Search Techniques: networks, *OR Spektrum*, (1989) 131-141.

Westin, C.-F., Mainer, S.E., Mamata, H., Nabavi, A., Jolesz, F.A., Kikinis, R. (2002). Processing and visualization for diffusion tensor MRI, *Medical Image Analysis*, Vol. 6 (2002) 93–108.

Yen, K. & Hanzo, L. (2000). Hybrid GA-based multiuser detection schemes for synchronous CDMA systems, *Proc. Vehicular Technology Conf.*, pp. 1400-1404, Tokyo, Japan, 2000.

Yen, K. & Hanzo, L. (2001). Genetic Algorithm assisted joint multiuser symbol detection and fading channel estimation for synchronous CDMA systems, *IEEE J. on Sel. Areas in Comm.*, Vol. 19, No. 6 (2001) 985–997.

7

Hybrid Tabu Algorithm for the Synthesis and Fabrication of Fiber Bragg Gratings

Nam Quoc Ngo[1] and Ruitao Zheng[2]
[1]*Nanyang Technological University*
[2]*Avago Technologies Manufacturing (Singapore) Pte. Ltd.*
Singapore

1. Introduction

Fiber Bragg gratings (FBGs) are widely employed as optical filters for performing various functions such as add/drop multiplexers, dispersion compensators and multiplexers/demultiplexers for use in optical communication systems and optical sensors because of a number of advantages that include low insertion loss, low polarization sensitivity, all-fiber geometry, compactness, easy fabrication and low cost (Kashyap, 1999; Othonos & Kalli, 1999). In addition, the technology of ultraviolet (UV) photoinduced FBGs is quite mature to allow the fabrication of a wide variety of FBGs with complex characteristics. To meet the increasing demand for large capacity of the next generation of optical communication systems (i.e., wavelength division multiplexing (WDM) networks) and optical sensors, there is an important need for a powerful design tool that can be used for the synthesis or design of FBGs from the specified frequency responses that can be practically realized. This synthesis or inverse problem of determining a FBG structure from a given frequency response (i.e., magnitude and phase responses) is common in many application areas. The design tool must be efficient and reliable to enable the synthesis of FBG-based filters with prescribed frequency responses, depending on the application requirements. That is, the design tool must be able to determine the index modulation profile and hence the structure of an FBG from a given frequency response. In addition, the design tool must be powerful enough for use in the diagnosis or characterization during and after fabrication of an FBG. Although several synthesis methods such as those based on the layer-peeling algorithm have been proposed for the synthesis of FBGs from the specified frequency responses, the index modulation profiles of the synthesized FBGs are not optimized and are often complex, making practical realization difficult (Feced et al., 1999; Poladian, 2000; Skaar et al., 2001; Rosenthal & Horowitz, 2003). To overcome this problem, optimization methods have been demonstrated as an attractive approach because it allows weighting mechanisms to be incorporated into the desired frequency response of an FBG-based filter to be synthesized, resulting in an optimum and practically realizable index modulation profile of the FBG structure. Furthermore, the optimization algorithms also allow additional constraints to be included in the weighting mechanisms of the specified frequency response to suit certain condition(s) or constraint(s) of a particular fabrication system. In optimization, the FBG synthesis problems are formulated as nonlinear objective

functions and the optimized solution of the FBG design is obtained by finding the global optimum of the objective function. Although several global optimization algorithms such as the global genetic algorithm (GA) (Skaar & Risvik, 1998; Gill et al., 2004; Cheng & Lo, 2004), the global simulated annealing (SA) (Dong, Azana & Kirk, 2003) and the local optimization method such as the Levenberg-Marquardt algorithm (Plougmann & Kristensen, 2004) have been applied to solving the FBG synthesis problems, the obtained solutions are, in general, not optimum, making practical implementation difficult. This is because, in general, it is important to employ a global optimization algorithm to solve an FBG synthesis problem to ensure that a global optimum can be obtained; however, convergence of the global optimization method is normally not as good as that of a local optimization algorithm. Unfortunately local optimization methods also cannot easily solve the FBG synthesis problems due to the multimodal and ill-conditioned character of the nonlinear objective functions. To improve the convergence of the global optimization algorithm, a hybrid algorithm combining a global optimization algorithm and a local optimization algorithm has been shown to be a better approach for the synthesis and fabrication of FBG-based bandpass filters (Zheng et al., 2004; Ngo et al., 2007). This is the motivation of this chapter which describes the use of a hybrid Tabu algorithm for the synthesis and fabrication of FBG-based bandpass filters and linear phase filters from the given frequency responses. (Ngo et al., 2004; Zheng et al., 2005) presented the first reports of employing the standard and improved Tabu search algorithms (Glover & Laguna, 1998; Chelouah & Siarry, 2000) for the synthesis of FBG-based bandpass filters and linear phase filters. The hybrid Tabu algorithm is a two-tier search that employs a global optimization algorithm (i.e., a staged continuous Tabu search (SCTS) algorithm (Zheng et al., 2005) which performs better than the standard Tabu search (Ngo et al., 2004)) and a local optimization algorithm (i.e., the Quasi-Newton method (Shanno, 1970) which has high efficiency in solving multimodal nonlinear optimization problems). First, the global SCTS algorithm, in which a dynamic mechanism for weighting of different requirements of the magnitude and phase responses is employed to enhance the optimization efficiency, is used to find a "promising" FBG structure that has a frequency response as close as possible to the target one. The local Quasi-Newton algorithm is then applied to further optimize this "promising" FBG structure obtained from the global SCTS algorithm to obtain the final optimum solution. To demonstrate the effectiveness of the hybrid Tabu method which has high convergence rate and high reliability, the synthesis and fabrication of several FBG-based bandpass filters and linear phase filters for application in optical communications are presented in this chapter. It is worth mentioning that other stochastic search algorithms of various types have also been applied to the design of FBGs: the Nelder-Mead Simplex hill climbing algorithm (Caucheteur et al., 2004) and particle swarm optimization (Baskar et al., 2005a), evolutionary strategies such as the covariance matrix adapted evolution (Baskar et al., 2005b; Baskar et al., 2006), and a multi-objective evolutionary algorithm (Manos & Poladian, 2005).

2. Transfer matrix method for solving non-uniform Bragg gratings

2.1 Theory of fiber Bragg grating

There are two main types of fiber Bragg gratings (FBGs), namely, uniform FBG (which has equal grating periods) and non-uniform FBG (which has unequal grating periods). Single-moded FBGs are considered here because they are commonly used in many areas of optics and photonics. For ease of discussion, the uniform FBG is considered in this section. An FBG

is essentially a filter written into the core of a segment of optical fiber via the interference of two ultraviolet (UV) beams from a UV laser (see Fig. 1). The interference pattern forms a periodic refractive index change (or index perturbation) along the longitudinal direction of the fiber. Due to the index change, the FBG acts as a series of reflectors, reflecting back a small amount of the input light with wavelength components that are close to the Bragg wavelength.

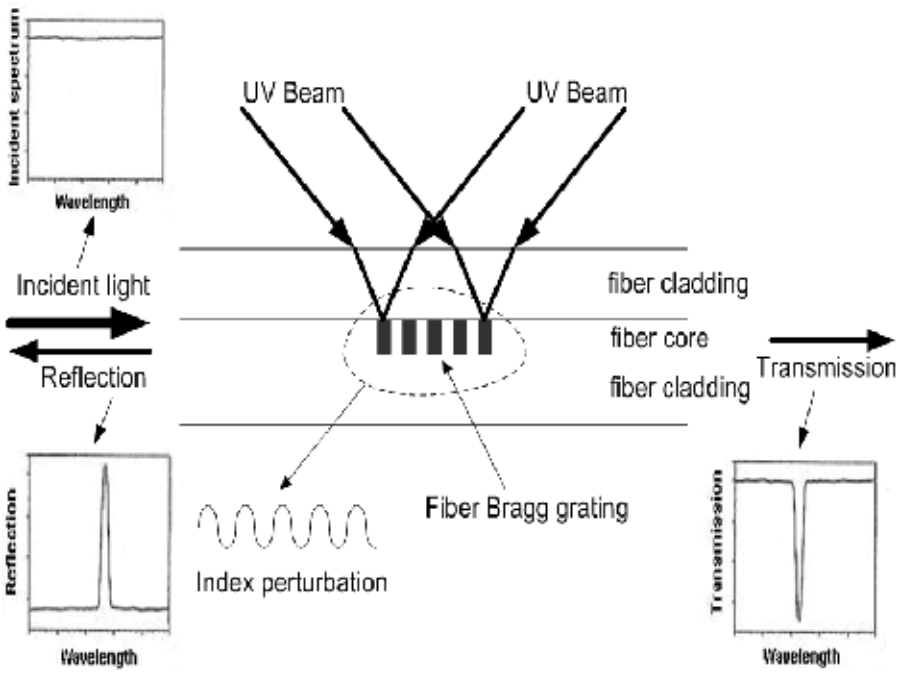


Fig. 1. Schematic diagram of a typical interferometric system used for the fabrication of fiber Bragg gratings (FBGs). The FBG shown here is of a uniform type with equal grating periods.

The Bragg wavelength, $\lambda_B(z)$, of a uniform FBG is the wavelength that fulfills the Bragg condition:

$$\lambda_B(z) = 2\bar{n}_{\text{eff}}(z)\Lambda(z) \tag{1}$$

where z is the longitudinal coordinate along the length of the grating and $\Lambda(z)$ is the perturbation period or grating period (see Fig. 2). The average effective refractive index on the grating is defined as (see Fig. 2)

$$\bar{n}_{\text{eff}}(z) = n_0 + \Delta n_{\text{dc}}(z) \tag{2}$$

where n_0 is the effective index without UV exposure and $\Delta n_{\text{dc}}(z)$ is the "dc" (or average) index change spatially averaged over the grating. Fig. 2 shows the quasi-sinusoidal profile of the effective index, $n_{\text{eff}}(z)$, which is described by

$$n_{\text{eff}}(z) = \bar{n}_{\text{eff}}(z) + \Delta n_{\text{ac}}(z) \cdot f(z) \cdot \cos\left[\frac{2\pi z}{\Lambda(z)} + \phi(z)\right] \tag{3}$$

where $\Delta n_{\text{ac}}(z)$ is the "ac" index change (i.e., index modulation), $f(z)$ is the normalized apodization function, and $\phi(z)$ is the chirp profile of the grating. Putting Eq. (2) into Eq. (3), the index perturbation, $\delta n_{\text{eff}}(z)$, is given by

$$\delta n_{\text{eff}}(z) = n_{\text{eff}}(z) - n_0 = \Delta n_{\text{dc}}(z) + \Delta n_{\text{ac}}(z) \cdot f(z) \cdot \cos\left[\frac{2\pi z}{\Lambda(z)} + \phi(z)\right] \tag{4}$$

Thus Eq. (4) shows that the optical properties of an FBG are essentially determined by the variation of the index perturbation along the grating length.

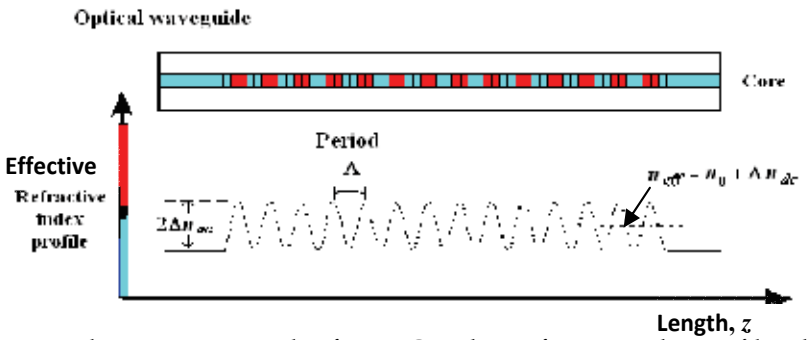


Fig. 2. Schematic showing an example of an FBG and its refractive index profile. The FBG shown here is of a uniform type with equal grating periods.

Figure 3 shows the index perturbation profiles of a uniform FBG, an apodized FBG with variable-dc index change, and an apodized FBG with zero-dc index change, which are considered in this work. Other types of index perturbation profiles such as chirped FBG, phase-shifted FBG and super-structured FBG can also be employed, depending on the desired filter responses (Erdogan, 1997).

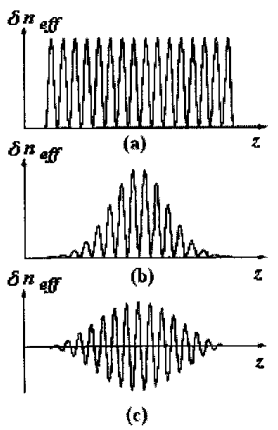


Fig. 3. Index modulation profiles of the types of FBGs considered here. (a) A uniform FBG with a constant dc index change. (b) An apodized FBG with variable-dc index change. (c) An apodized FBG with zero-dc index change.

The length of the FBG typically ranges from several millimeters to tens of centimeters, depending on the desired filter responses. Because there are tens of thousands of these perturbation periods of index changes or reflectors in a row, an FBG-based filter generally has a near-squared reflective magnitude response. In the reflection mode, the reflective magnitude response has a very narrow bandpass response (with a typical 3-dB bandwidth of 0.2 nm or 25 GHz in the 1550 nm wavelength window) at the Bragg wavelength; whereas in the transmission mode, the transmissive magnitude response has a very narrow notch response at the Bragg wavelength (see Fig. 1). The phase-mask technique is one of the most effective methods used in the fabrication of FBGs because it employs a simple diffractive optical element (or a phase mask) to spatially modulate the UV beam (Kashyap, 1999; Othonos & Kalli, 1999). Compared with two other main types of optical filters, namely, thin film filters and arrayed waveguide gratings, FBG-based filters have many unique advantages such as low loss, low polarization sensitivity, simple all-fiber geometry, easy fabrication, and low cost.

2.2 Transfer matrix method for solving non-uniform fiber Bragg gratings

The coupled-mode theory has been widely used for the analysis of FBGs because it allows one to determine the effect of the grating structure on the frequency response (i.e., magnitude and phase responses) (Erdogan, 1997; Hill & Meltz, 1997; Kashyap, 1999; Othonos & Kalli, 1999). A uniform FBG is the simplest type of FBG to design and fabricate because it simply has equal grating periods and a constant modulation depth of the refractive index (see Fig. 1, Fig. 2, Fig. 3(a)). The uniform FBG can be easily designed because an analytical solution to the coupled-mode equation can be easily obtained. However, due to the finite length of the uniform FBG, the roll offs on the two edges of the reflective magnitude response are not sharp enough due to the presence of sidelobes on both sides of the bandpass response. This drawback has limited the application of uniform FBGs. This limitation can be overcome by using a non-uniform FBG which has a more squared reflective magnitude response with much smaller amplitudes of the sidelobes. However, analytical solutions to the coupled-mode equations describing the non-uniform FBGs cannot be easily obtained. The transfer matrix method (TMM) has been widely employed for solving the non-uniform FBGs due to its high computational efficiency and high reliability (Erdogan, 1997; Kashyap, 1999). The TMM method allows the magnitude and phase responses of a non-uniform FBG to be easily obtained with reasonably high accuracy. In TMM, a non-uniform FBG can be modeled using two methods, depending on the requirements of the magnitude and phase responses: (1) A non-uniform FBG model using the cascade of serially-connected *uniform* sub-gratings (this model is sufficient for obtaining the magnitude response when the phase response is not important); and (2) A non-uniform FBG model using the cascade of serially-connected *apodized* sub-gratings (this model must be used when both the magnitude and phase responses are important). These two methods are described below.

2.2.1 A non-uniform FBG model using the cascade of *uniform* sub-gratings

When only the magnitude response of a filter (whose phase response is not important) is required to be designed, the TMM method can be used to model a non-uniform FBG as the cascade of serially-connected *uniform* sub-gratings. The non-uniform FBG can be divided into a number of serially-connected *N uniform* sub-gratings or sections (as shown in Fig. 4).

Each *uniform* sub-grating section can be described by an analytic transfer matrix. The transfer matrix for the entire non-uniform FBG structure can be obtained by simply multiplying the individual transfer matrices of the *uniform* sub-gratings.

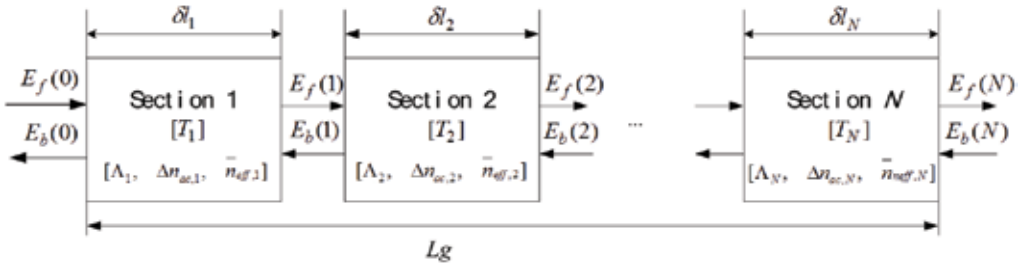


Fig. 4. Schematic diagram of a non-uniform FBG model based on the cascade of serially-connected *uniform* sub-gratings.

Each *uniform* sub-grating section is described by an index perturbation defined in Eq. (4). Note that Eq. (4) is a general equation describing the index perturbation of all kinds of FBGs. To apply Eq. (4) to each *uniform* sub-grating section, the parameters in this equation are re-defined as follows. A *uniform* sub-grating section requires $f(z)=1$ and $\phi(z)=0$ in Eq. (4). Putting these conditions into Eq. (4) gives a simpler index perturbation, $\delta n_{\text{eff}}(z)$, for each *uniform* sub-grating which is given by

$$\delta n_{\text{eff}}(z) = \Delta n_{\text{dc}}(z) + \Delta n_{\text{ac}}(z)\cdot\cos\left(2\pi z/\Lambda(z)\right) \tag{5}$$

Equation (5) is schematically shown in Fig. 3(a). In Fig. 4, $E_{\text{f}}(j;\lambda)$ and $E_{\text{b}}(j;\lambda)$ are the complex electric fields of the forward and backward propagation waves, respectively, describing the j^{th} section. Also, δl_j, Λ_j, $\Delta n_{\text{ac},j}$ and $\bar{n}_{\text{eff},j}$ (and hence $\Delta n_{\text{dc},j}$ according to Eq. (2)), which are the parameters to be optimized in this particular model, are the length, period, 'ac' index change (i.e., index modulation) and average effective index of the j^{th} section, respectively. In this model, the parameters that are not optimized are Λ_j and $\Delta n_{\text{dc},j}(z)=\Delta n_{\text{dc},j}$ (and hence $\bar{n}_{\text{eff},j}$ according to Eq. (2)), which are the period and the "dc" index change, respectively, and they are fixed or constant values. L_{g} is the total length of the non-uniform FBG. The designer has the choice of choosing which of these four (4) variables (i.e., δl_j, Λ_j, $\Delta n_{\text{ac},j}$ and $\bar{n}_{\text{eff},j}$) are to be optimized, depending on the fabrication condition(s) or constraint(s) of a particular fabrication system. The complex electric fields at the input ports ($E_{\text{f}}(0;\lambda)$, $E_{\text{b}}(0;\lambda)$) and output ports ($E_{\text{f}}(N;\lambda)$, $E_{\text{b}}(N;\lambda)$) of the non-uniform FBG are described by

$$\begin{bmatrix} E_{\text{f}}(0;\lambda) \\ E_{\text{b}}(0;\lambda) \end{bmatrix} = T\cdot\begin{bmatrix} E_{\text{f}}(N;\lambda) \\ E_{\text{b}}(N;\lambda) \end{bmatrix}; \qquad T=\prod_{j=1}^{N} T_j \tag{6}$$

where $T_j=\begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix}$ is the 2 × 2 transfer matrix of the j^{th} section. The elements of the transfer matrix are defined as

$$T_{11} = \cosh(S_j \delta l_j) - \frac{i\hat{\sigma}_j}{S_j}\sinh(S_j \delta l_j), \quad T_{22} = \cosh(S_j \delta l_j) + \frac{i\hat{\sigma}_j}{S_j}\sinh(S_j \delta l_j) \tag{7}$$

$$T_{12} = -\frac{i\kappa_j}{S_j}\sinh(S_j \delta l_j), \quad T_{21} = \frac{i\kappa_j}{S_j}\sinh(S_j \delta l_j) \tag{8}$$

where $i = \sqrt{-1}$. The 'ac' coupling coefficient, κ_j, is defined as

$$\kappa_j = i\frac{\pi}{\lambda}\Delta n_{\mathrm{ac},j} \tag{9}$$

where λ is the optical wavelength. The 'dc' coupling coefficient, $\hat{\sigma}_j$, is defined as

$$\hat{\sigma}_j = \delta_j + \sigma_j \tag{10}$$

where $\delta_j = \beta_j - \pi/\Lambda_j = 2\pi\bar{n}_{\mathrm{eff},j}\left(1/\lambda_j - 1/\lambda_{\mathrm{B},j}\right)$ is the detuning parameter at the wavelength λ_j, β_j is the propagation constant, $\lambda_{\mathrm{B},j} = 2\bar{n}_{\mathrm{eff},j}\Lambda_j$ is the Bragg wavelength, and $\bar{n}_{\mathrm{eff},j} = n_0 + \Delta n_{\mathrm{dc},j}$ (see Eq. (2)). The coefficient σ_j is defined as

$$\sigma_j = \frac{2\pi}{\lambda_j}\Delta n_{\mathrm{dc},j} \tag{11}$$

where $\Delta n_{\mathrm{dc},j}$ is given in Eqs. (2) and (4). S_j in Eqs. (7) and (8) is defined as

$$S_j^2 = \kappa_j^2 - \hat{\sigma}_j^2 \tag{12}$$

Applying the boundary condition $E_{\mathrm{b}}(N;\lambda) = 0$ (i.e., there is no input to the right side of the FBG), the reflection frequency response $\rho(\lambda)$ and the transmission frequency response $t_x(\lambda)$ are given by

$$\rho(\lambda) = \frac{E_{\mathrm{b}}(0;\lambda)}{E_{\mathrm{f}}(0;\lambda)} \tag{13}$$

$$t_x(\lambda) = \frac{E_{\mathrm{f}}(N;\lambda)}{E_{\mathrm{f}}(0;\lambda)} \tag{14}$$

The reflection magnitude response and the transmission magnitude response are simply given by $|\rho(\lambda)|^2$ and $|t_x(\lambda)|^2$, respectively. The reflection phase response and the transmission phase response are defined as $\arg[\rho(\lambda)]$ and $\arg[t_x(\lambda)]$, respectively, where arg stands for argument. We here consider only the reflection frequency response (i.e., magnitude and phase responses) or Eq. (13) because we are interested in the bandpass

response of the filter. The transmission magnitude response will give a bandstop or notch response. The delay time, $\tau_\rho(\lambda)$, of light reflected off a grating corresponds to the phase change of $\rho(\lambda)$ relative to the optical wavelength λ, and is given by (Erdogan, 1997)

$$\tau_\rho(\lambda) = -\frac{\lambda^2}{2\pi c} \cdot \frac{d\theta_\rho}{d\lambda} \tag{15}$$

where $\theta_\rho = \arg\left[\rho(\lambda)\right]$ is the phase response of $\rho(\lambda)$ and c is the speed of light in vacuum. The dispersion of the grating, $d_\rho(\lambda)$, is therefore given by

$$d_\rho(\lambda) = \frac{d\tau_\rho}{d\lambda} = -\frac{\lambda^2}{2\pi c}\left(\frac{d^2\theta_\rho}{d\lambda^2} + \frac{2}{\lambda} \cdot \frac{d\theta_\rho}{d\lambda}\right) \tag{16}$$

The non-uniform FBG model based on the cascade of serially-connected *uniform* sub-gratings described here is useful for use in the design of the magnitude response (but not the phase response) of a bandpass filter. The model will be used for the design and fabrication of bandpass filters (where the phase responses are not of interest) which are described in Section 4.3. When both the magnitude and phase responses of the filter (e.g., a linear phase filter) are required to be designed, the non-uniform FBG model based on the cascade of serially-connected *apodized* sub-gratings must be used and it is described below.

2.2.2 A non-uniform FBG model using the cascade of *apodized* sub-gratings

When both the magnitude and phase responses of a filter are required to be designed, the TMM method can be used to model a non-uniform FBG as the cascade of serially-connected *apodized* sub-gratings. The non-uniform FBG can be divided into a number of serially-connected *N apodized* sub-gratings or sections (as shown in Fig. 5). Each *apodized* sub-grating section can be described by an analytic transfer matrix. The transfer matrix for the entire non-uniform FBG structure can be obtained by simply multiplying the individual transfer matrices of the apodized sub-gratings.

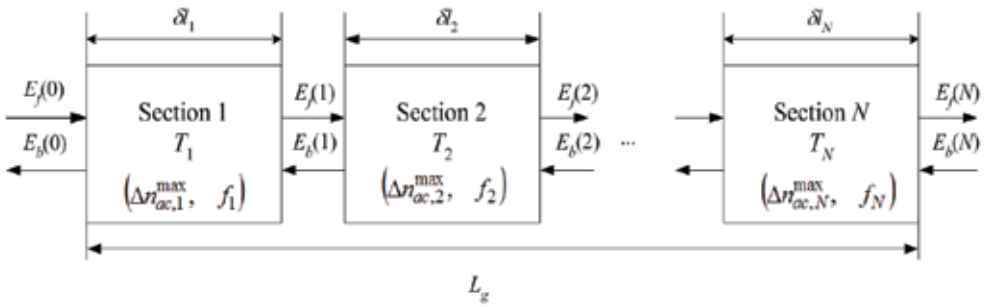


Fig. 5. Schematic diagram of a non-uniform FBG model based on the cascade of serially-connected *apodized* sub-gratings.

Each *apodized* sub-grating section is described by the index perturbation defined in Eq. (4). Note that Eq. (4) is a general equation describing the index perturbation of all kinds of FBGs. To apply Eq. (4) to each *apodized* sub-grating section, the parameters in this equation are re-

defined as follows. An *apodized* sub-grating section requires $\phi(z) = 0$ in Eq. (4). Putting this condition into Eq. (4) gives a simpler index perturbation, $\delta n_{\text{eff}}(z)$, for each *apodized* sub-grating which is given by

$$\delta n_{\text{eff}}(z) = \Delta n_{\text{dc}}(z) + \Delta n_{\text{ac}}(z) \cdot f(z) \cdot \cos\left(2\pi z/\Lambda(z)\right) \tag{17}$$

Equation (17) is schematically shown in Fig. 3(b). In Fig. 5, $E_{\text{f}}(j;\lambda)$ and $E_{\text{b}}(j;\lambda)$ are the complex electric fields of the forward and backward propagation waves, respectively, describing the j^{th} section. Also, δl_j, $\Delta n_{\text{ac},j}$ and f_j, which are the parameters to be optimized in this particular model, are the length, 'ac' index change (i.e., index modulation) and apodization function of the j^{th} section, respectively. In this model, the parameters that are not optimized are Λ_j and $\Delta n_{\text{dc},j}(z) = \Delta n_{\text{dc},j}$ (and hence $\bar{n}_{\text{eff},j}$ according to Eq. (2)), which are the period and the "dc" index change, respectively, and they are fixed or constant values. L_{g} is the total length of the non-uniform FBG. The designer has the choice of choosing which of these five (5) variables (i.e., δl_j, $\Delta n_{\text{ac},j}$, f_j, Λ_j and $\Delta n_{\text{dc},j}$) are to be optimized, depending on the fabrication condition(s) or constraint(s) of a particular fabrication system. Unlike a uniform FBG (i.e., no apodization) which has sidelobes of about -10 dB in the reflective magnitude response, applying apodization to a non-uniform FBG can significantly suppress the sidelobes to > -40 dB in the reflective magnitude response. Listed below are several commonly used apodization functions which can be applied to each sub-grating section

$$\text{Raised cosine: } f_j(z) = \frac{1}{2}\left\{1 + \cos\left[\frac{\pi\left(z - \delta l_j/2\right)}{\delta l_j}\right]\right\} \tag{18}$$

$$\text{Gaussian: } f_j(z) = \exp\left[-\frac{\left(4\ln 2\right)\left(z - \delta l_j/2\right)^2}{\left(\delta l_j/3\right)^2}\right] \tag{19}$$

$$\text{Sine: } f_j(z) = \sin\left(\pi z/\delta l_j\right) \tag{20}$$

$$\text{Quadratic sine: } f_j(z) = \sin^2\left(\pi z/\delta l_j\right) \tag{21}$$

Note that the only difference between a *uniform* sub-grating (as described in Section 2.2.1) and an *apodized* sub-grating is that the former has an index perturbation described by Eq. (5) while the latter has an index perturbation described by Eq. (17). In general, the number of *apodized* sub-gratings is smaller than the number of *uniform* sub-gratings to achieve a particular desired frequency response. Thus Eqs. (6)–(12) are also applicable to the *apodized* sub-gratings by putting the right condition defined in Eq. (17) into Eqs. (6)–(12). Putting Eq. (17) into Eqs. (6)–(12), the reflection frequency response, $\rho(\lambda)$, and the transmission frequency response, $t_x(\lambda)$, can be computed using Eqs. (13) and (14), respectively. Also, the

delay time, $\tau_\rho(\lambda)$, of light reflected off a grating can be determined using Eq. (15) and the dispersion of the grating, $d_\rho(\lambda)$, can be calculated using Eq. (16). This model will be used for the design of FBG-based linear phase filters which is described in Section 4.4.

3. Staged continuous Tabu search algorithm

3.1 Review of Tabu search algorithm

Tabu search (TS) is an iterative search method originally developed by Glover and Laguna (Glover & Laguna, 1998) which has been successfully applied to a variety of combinatorial global optimization problems (Ho et al., 2001; Machado et al., 2001; Chelouah & Siarry, 2000). The TS algorithm has been used for the synthesis of FBG-based linear phase filters using the non-uniform FBG model based on the cascade of serially-connected *apodized* sub-gratings (Ngo et al., 2004). A good analogy of how TS works is mountain climbing, where the climber must selectively remember key elements of the path traveled (using *adaptive memory*) and must be able to strategize choices along the way (using *responsive exploration*). A rudimentary form of this algorithm can be summarized as follows. It starts from an initial solution s that is randomly selected. From this current solution s, a set of neighbors, called s', is generated by pre-defining such a set of "moves" or perturbations of the current solution. To avoid the endless reiterative cycle, the neighbors of the current solution, which belong to a subsequently defined "tabu list", are systematically eliminated. The objective function to be minimized is then evaluated for each generated solution s', and the best neighborhood of s becomes the new current solution s' even if it is worse than s. The "move" that generates the new current solution will also be stored in the tabu list, which is circular. When the tabu list is full, it is updated by eliminating the previous estimated solution. Then a new "iteration" is performed; the previous procedure is repeated by starting from the new current solution until a pre-defined stopping condition is satisfied. The TS algorithm has a small probability of becoming trapped in a local optimum. Compared to other traditional methods such as the genetic algorithm (GA) and the simulated annealing (SA), the TS algorithm has one unique advantage in that it can also be organized to take advantage of problem-specific information and thus has higher convergence velocity as well as higher level of reliability. TS also includes "candidate list strategies" for generating and sampling neighbors. These candidate list strategies are very important because often only a relatively small subset of neighbors is generated at any given iteration, especially when a large number of neighborhoods is used, as in the case of multi-variable problems whose neighbors are generated in a multi-dimensional space.

3.2 Review of continuous Tabu search algorithm

As an enhancement to the TS algorithm described above in terms of higher convergence velocity and higher level of reliability, a continuous Tabu search (CTS) algorithm, which employs a special "candidate list strategy" to generate neighbors, has been proposed for the optimization of nonlinear objective functions (Siarry & Berthiau, 1997). In this method, the solution space is divided into several regions. Neighbors are generated in these regions and the remainder of the method consists of an elementary form of TS that uses only the simple tabu list construction as described in Section 3.1. A brief review of the CTS algorithm is described here because it will be used for the development of a staged continuous search (SCTS) algorithm in the subsequent section. For the following optimization problem:

$$\min_{s\in\Psi^k}\left[\Phi(s)\right], \tag{22}$$

where $\Phi(s)$ is the objective function to be minimized, and $s=[x_1,x_2,\cdots,x_k]^{\mathrm{T}}$ is defined as

$$s\in\Psi^k \text{ and } \Psi^k=\{s\,|\,a_j\le x_j\le b_j\},\ j=1, 2, \ldots, k. \tag{23}$$

where a_j and b_j are the boundary values of the j^{th} element (or x_j) of s, and k represents the dimension of the problem or the number of variables. The basic process of the CTS method, which is organized around a simple version of the tabu search, can be summarized as follows.

1. Generate a random point s that belongs to the space Ψ^k as the current solution.
2. A set of neighbors, $s'\in\Psi^k$, is then generated by applying s with a series of perturbations or "moves". Generation of neighbors is defined as follows. The neighborhood space Ψ^k of the current solution, s, is deemed as a ball $B(s,r)$ centered on s with a radius r. Considering a set of concentric balls with radii h_0, h_1, …, h_n, the space is partitioned into n concentric 'crowns'. Hence n neighbors of s are obtained by selecting one point randomly inside each crown and eliminating those neighbors that belong to the "tabu list". Figure 6 shows an example that demonstrates the generation of neighbors for a problem with two variables x_1 and x_2 (k = 2), where the space is partitioned into four (n = 4) concentric "crowns", and four neighbors are produced randomly in their own crown areas.
3. Evaluate these neighbors with the objective function, choose the best neighbor s^* and replace the starting point s with s^* even if it is worse than the current solution. Then update the "tabu list".
4. Clear the "tabu list": in particular those solutions that belong to the "tabu list" can release their tabu status if their "aspiration levels" are sufficiently high.
5. Check the stopping condition and return to step 2 if the condition is not met. Otherwise, stop the iteration procedure and report the results.

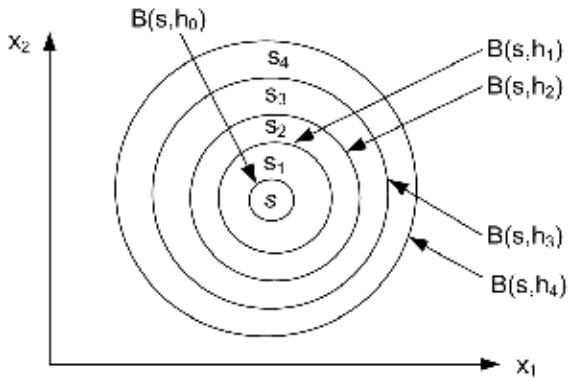


Fig. 6. Partition of the current solution neighborhood, where x_1 and x_2 are two variables (k = 2) and n = 4 concentric "crowns". The neighborhood s_j (j = 1, 2, 3, 4) is selected randomly in its own crown area (Siarry & Berthiau, 1997).

Figure 7 shows the flow chart of the CTS algorithm, where the main stages include the initial solution, generation of neighbors, selection of the solution and tabu list clearance. The strategy of generating neighbors in CTS is more efficient than a naive candidate-list strategy based solely on random sampling, and usually produces neighbors distributed over the whole solution space. Although the method is effective for optimizing functions with two or three variables, its efficiency decreases with an increase in the number of variables of a function such as the case for high-dimension problems. These difficulties can be overcome using a staged continuous Tabu search (SCTS) algorithm which is described next.

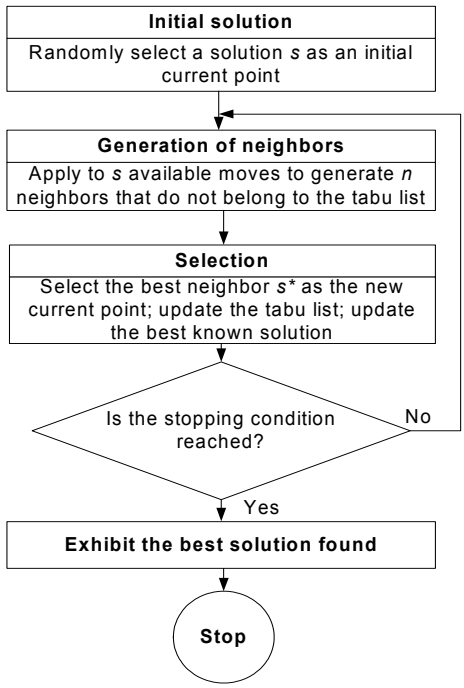


Fig. 7. General flow chart of a continuous Tabu search (CTS) algorithm.

3.3 Staged continuous Tabu search algorithm

The staged continuous Tabu search (SCTS) algorithm employs the same rudimentary form of tabu search embodies in the CTS algorithm (Zheng et al., 2005). However, SCTS provides an enhanced candidate-list strategy that subdivides the CTS approach into three independent processes that generate candidate neighbors in a different way (see Fig. 8). The first stage attempts to survey the whole solution space to localize a "prospective point", which is a solution likely to produce a global optimum. The objective of the second stage is to find a point close to the global optimum. The third stage starts from the solution found in the second stage, and eventually converges to the global optimum point. The SCTS algorithm is described below with reference to Fig. 8 which shows the steps of the algorithm. The following notations are used in Fig. 8: Ψ^k: space of feasible solutions (k dimensions); s_0 : current solution; n_1 : length of neighbors generated in the first stage (which

is equal to k here); n_2 : section number divided within the boundary of every element of s_0; n_3 : length of neighbors generated in the third stage; s': the neighborhood of s_0; s^*: the best solution in s'; s_{opt} : current best solution found; and $Mv(j)$: maximum number of iterations without improvement of s_{opt} in the j^{th} stage.

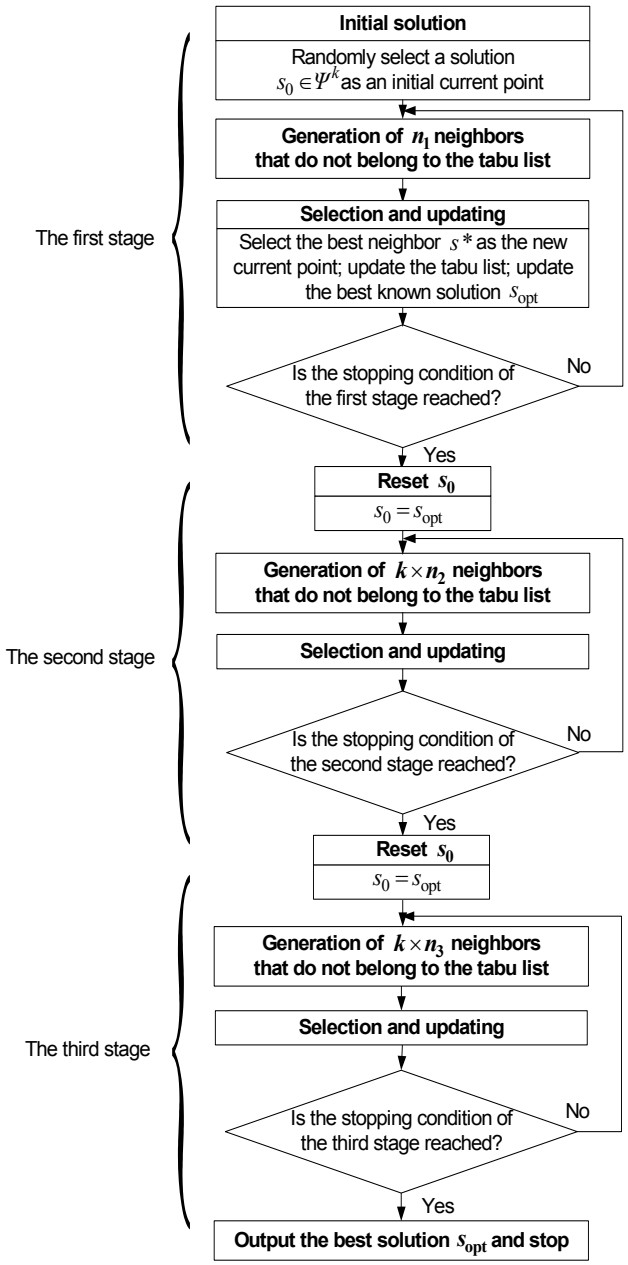


Fig. 8. Algorithmic description of the SCTS algorithm.

Generation of Neighborhoods: Neighborhoods are generated in a ball $B(s,r)$ centered on s with a radius r (see Fig. 6). All neighbors s' meet the condition: $|s'-s| \le r$. In the first stage, the radius r_1 is defined so that the ball $B_1(s,r_1)$ contains the whole k-dimension space Ψ^k. With radii $r_1^1, r_1^2, \cdots, r_1^{n_1}$, the ball is partitioned into k concentric "crowns" centered on the current solution. One neighbor is produced in each crown. Thus the j^{th} neighbor, s_j', is generated with the condition

$$r_1^{j-1} \le |s_j' - s| \le r_1^j, \quad (r_1^0 = 0). \tag{24}$$

As the ball $B_1(s,r_1)$ includes the whole space Ψ^k, it should be possible for all solutions within it to become neighbors of the current solution, s, so that the process can investigate the whole solution space. We define the "moves" to generate neighbors such that some elements of the current solution are randomly replaced. The number of replaced elements depends on different crowns. For example, the j^{th} neighbor, s_j', is generated by replacing any j element of the current solution. The radius r_2 for the generation of neighbors in the second stage is defined as the minimum radius of radii $r_1^1, r_1^2, \cdots, r_1^{n_1}$ defined in the first stage. Followed with another partition process with a set of radii $r_2^1, r_2^2, \cdots, r_2^{n_2}$, the ball $B_2(s,r_2)$ is divided into n_2 sections. The j^{th} neighbor, s_j', is generated with a condition given by

$$r_2^{j-1} \le |s_j' - s| \le r_2^j, \quad (r_2^0 = 0). \tag{25}$$

The minimum radius defined in the first stage is propagated in only one dimension of the current solution. Considering the condition defined in Eq. (23), the boundary can be proportionally divided for every dimension into n_2 partitions. The neighbors can then be generated by replacing the j^{th} element of the current solution, x_j', with a number computed by:

$$x_j' = a_j + (j_2 + \mu) \times \frac{b_j - a_j}{n_2}, \quad \text{where } j = 1, 2, \ldots, k;\ j_2 = 1, 2, \ldots, n_2; \tag{26}$$

where μ is a random value between 0 and 1, a_j and b_j are defined in Eq. (23), and $k \times n_2$ is the number of neighbors in the stage. The minimum of radii $r_2^1, r_2^2, \cdots, r_2^{n_2}$ is set as radius r_3 to generate neighbors in the third stage. Instead of partitioning to generate neighbors, the radius of the ball $B_3(s,r_3)$ decreases with an increase in the iteration number. The generation of the j^{th} neighbor is defined as

$$x_j' = x_j + \mu_j \times \frac{b_j - a_j}{n_2} \times \frac{M_3 - m}{M_3} \tag{27}$$

where x_j and x_j' are the j^{th} elements of the current solution s and the neighbor s' produced, respectively, m is the iteration number without any improvement on the current solution, and M_3 is the maximum allowable number of iterations without any improvement on the current solution. As pointed out previously, the neighbors in the first stage are generated in the largest possible range so as to explore most of the space. While in the last stage, only a reduced space is used so that the solution will eventually converge to the global optimum.

Tabu list: In the underlying TS algorithm, a tabu list stores some solutions that have recently been selected. It is used to qualify the algorithm to select solutions that have not been selected before so as to escape from being recycled. Because the three stages in the SCTS algorithm are independent of each other, the tabu lists in these stages are thus independent of each other. The list obtained in the first stage will store those 'prospective solutions' found in recent iterations. In the second and third stages, the lists will store the attributes of 'moves' or perturbations that generate the best neighbors in recent iterations. The tabu list in each stage is always reset at the beginning of each stage.

Stopping conditions: The stopping conditions for the three processes are defined below.

1. The program will stop after a given number of iterations when there is no improvement on the value of the objective function. The number of iterations varies in different stages.
2. The result satisfies the prescribed condition. An example of the prescribed condition is the known global optimum of a benchmark test function.
3. The search procedure will stop after it has completed a pre-defined maximum number of iterations.

All stages will be terminated if any one of these stopping conditions is satisfied. That is, if the process is in the first or second stage, it will move into the next stage. However, if the process is in the third stage, the algorithm will stop and report the result.

The sensitivities of several main parameters of the CTS algorithm were discussed (Siarry & Berthiau, 1997). Usually, these parameters should be adjusted empirically according to the nature of the problem so as to achieve an efficient optimization. As the SCTS algorithm is derived from the CTS algorithm, it is found that the properties of some parameters in the SCTS algorithm are similar to those of the CTS algorithm. These parameters are not analyzed individually, and instead a set of empirical values is applied in the experiments for testing the benchmark functions. These empirical values are listed in Table 1.

Parameters used in the SCTS algorithm	Parameters used in the benchmark functions
Number of neighbors in the first stage (n_1)	Number of variables
Number of sections in the second stage (n_2)	5
Number of neighbors in the third stage (n_3)	Number of variables
Number of neighbors in the second stage	5 × number of variables
Maximum number of iterations without any improvement on the objective function value (Mv)	{20, 8, 5} for {1st stage, 2nd stage, 3rd stage}
Maximum number of iterations of the SCTS algorithm	8000

Table 1. Parameters of the SCTS algorithm used for testing the benchmark functions.

Numerical experimental results: To demonstrate the effectiveness of the SCTS algorithm, the important parameters to be studied are convergence, speed and robustness. Convergence refers to the evaluation of the search for the global optimum of a function. The test for convergence employed here is the relative error between the optimum obtained by the algorithm, X_{opt}, and a theoretical (or known) value of the optimum, X_{theo}, of the function. The relative error, E_{relative}, is defined as (Andre et al., 2000)

$$E_{\text{relative}} = \frac{\left| X_{\text{opt}} - X_{\text{theo}} \right|}{X_{\text{theo}}} \tag{28}$$

If X_{theo} = 0, Eq. (28) is reduced to

$$E_{\text{relative}} = \left| X_{\text{opt}} - X_{\text{theo}} \right| \tag{29}$$

The criterion of speed refers to the time taken by the algorithm to find the global optimum of an objective function. As the computation time also depends on the speed of the computer, it is better to define the speed criterion by determining the number of evaluations of the objective function required till a global optimum is found. Robustness means that the algorithm is versatile and can be applied to solving a variety of functions. A set of commonly used benchmark functions with known global optima (as listed in the Appendix) is employed to test the algorithm. These benchmark test functions represent various practical problems in engineering. To obtain a statistical comparison of the optimization results, every test is performed 100 times (starting from various randomly selected points) to ensure that the results obtained are reliable.

Table 2 shows the results obtained from the CTS (Siarry & Berthiau, 1997) and SCTS algorithms for the four test functions with two and three variables. The criterion of success is the percentage of trials (out of 100 tests for each function) required to obtain the global optimum with a relative error of < 1%. From the table, both algorithms can successfully find the global optima of all the four test functions which are given in the Appendix. However, the number of evaluations of the Goldprice and Shubert test functions required by the SCTS algorithm (i.e., 696 and 521, respectively) is much smaller than those required by the CTS algorithm (i.e., 1636 and 1123, respectively). This means that the SCTS algorithm has a faster computation rate than that of the CTS algorithm for these two particular test functions. However, the two algorithms have about the same computation rate for the Hartmann1 and Branin functions.

Function	Success rate (%)		Number of evaluations of objective or test functions	
	CTS	SCTS	CTS	SCTS
Goldprice	100	100	1636	696
Hartmann1	100	100	528	691
Branin	100	100	668	491
Shubert	100	100	1123	521

Table 2. Experimental data of the SCTS and CTS algorithms.

Table 3 shows a comparison of the experimental data of various test functions obtained by the SCTS algorithm and an improved genetic algorithm (IGA). Note that the IGA algorithm can potentially yield a complete set of optima of multimodal problems (Andre, Siarry & Dognon, 2000). These test functions have variables ranging from 1 to 20 as given in the Appendix. In the table, Max and Min are, respectively, the maximum and minimum values of a set of optima found over 100 tests. The SCTS algorithm outperforms the IGA algorithm in two ways. The SCTS algorithm has one advantage in that it can find the global optima of the test functions (i.e., Brown1, Brown 3 and F10n functions) with 100% success rate and with very low relative errors; while the IGA algorithm has very low success rates and very high relative errors of these test functions. The other advantage of the SCTS algorithm over the IGA algorithm is that it has a much smaller computation time due to the smaller number of evaluations of the test functions. In addition, for 100% success rates of the test functions achieved by both algorithms, the SCTS algorithm has much smaller relative errors than those of the IGA algorithm. It can thus be concluded from the results shown in Tables 2 and 3 that the SCTS algorithm outperforms the CTS algorithm and the IGA algorithm in terms of higher success rate and higher computation efficiency. Next section describes the use of the SCTS algorithm for the development of a hybrid Tabu search algorithm which is a hybrid of the global SCTS algorithm and the local Quasi-Newton algorithm.

4. Hybrid Tabu search algorithm for optimization and fabrication of non-uniform fiber Bragg gratings

4.1 Staged continuous Tabu search algorithm for optimization of non-uniform fiber Bragg gratings

This section describes the SCTS algorithm (as described in Section 3.3) for use in the optimization of non-uniform FBGs, and it is schematically shown in Fig. 9.

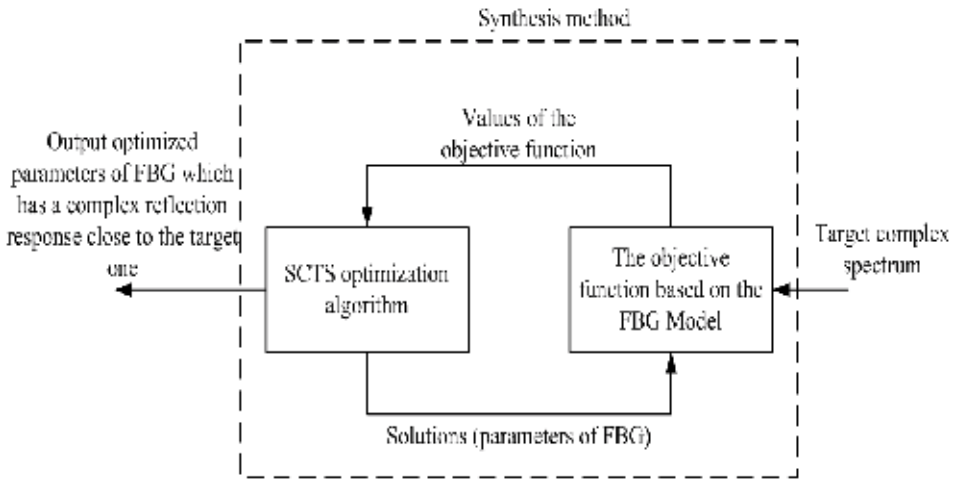


Fig. 9. Block diagram of the SCTS algorithm for use in the optimization of FBGs.

The synthesis problem of an FBG is formulated as an objective function which measures the error between the designed reflection frequency response of the FBG using the SCTS algorithm and the target frequency response. The synthesized FBG using the SCTS algorithm can be carried out using the steps as described below.

Function	Number of variables	Theoretical minimum value of optimum	Minimum value of a set of optima found (Min)		Maximum value of a set of optima found (Max)		Average of relative errors over 100 tests (%)		Number of evaluations of test functions		Success rate (%)	
			IGA	SCTS	IGA	SCTS	IGA	SCTS	IGA	SCTS	IGA	SCTS
F1	1	–1.1232	–1.1232	–1.1232	–1.1139	–1.1223	0.03	0	784	134	100	100
F3	1	–12.0312	–12.0312	–12.0312	–11.9270	–12.0203	0.12	0	744	181	100	100
Branin	2	0.3979	0.3979	0.3979	0.4018	0.3983	0.48	0	2040	492	100	100
Goldprice	2	3	3.003	3.002	3.0296	3.0029	0.43	0	4632	696	100	100
Shubert1	2	–186.7309	–186.6857	–186.7304	–184.9554	–186.3406	0.53	0.014	8853	2194	100	100
Shubert2	2	–186.7309	–186.7047	–186.7302	–184.9295	–185.9505	0.53	0.024	4116	2053	100	100
Shubert	2	–186.7309	–186.7280	–186.7269	–184.8753	–186.5490	0.49	0.0003	2364	521	100	100
Hartmann1	3	–3.8628	–3.8611	–3.8621	–3.8246	–3.8591	0.51	0.018	1680	560	100	100
Brown1	20	2	8.5516	2.0018	111.2914	2.0020	2692.67	0.05	128644	111430	0	100
Brown3	20	0	0.6746	0.0006	5.9122	0.0010	2.324	0.0009	106859	15142	5	100
F5n	20	0	0.0022	0.0001	0.5906	0.0010	0.067	0.0004	99945	17443	100	100
F10n	20	0	0.0496	0.0001	4.0660	0.0010	1.197	0.0008	113929	19931	49	100
F15n	20	0	0.0034	0.0003	0.7361	0.0009	0.075	0.0008	102413	19660	100	100

Table 3. Experimental data of the test functions obtained by the SCTS algorithm and the IGA algorithm.

1. When the target frequency response (or complex spectrum) is input into the synthesis method, the FBG model is formulated as an objective function to be minimized. The FBG model can either be based on the cascade of serially-connected *uniform* sub-gratings (Section 2.2.1) for the synthesis of the desired magnitude response (where the phase response is not important) or the cascade of serially-connected *apodized* sub-gratings (Section 2.2.2) for the synthesis of the desired frequency response (which includes both the magnitude and phase responses).
2. The algorithm will produce a set of solutions. These solutions include the index modulation profiles, $\Delta n_{\mathrm{ac},j}$ $(j=1,2,\cdots M)$, of the cascade of *uniform* sub-gratings (Eq. (5)) or the cascade of the *apodized* sub-gratings (Eq. (17)).
3. These solutions are sent to the objective function built in step (1). The values of the objective function are calculated and sent back to the algorithm.
4. The algorithm checks the calculated values of the objective function. If the pre-defined stopping conditions are not reached, the process will enter the next iteration and will be recycled from step (2). Otherwise, the algorithm will output the best solution found, that is, the optimized structural parameters (δl_j, $\Delta n_{\mathrm{ac},j}$, f_j, Λ_j and $\Delta n_{\mathrm{dc},j}$) of the designed FBG.

The SCTS algorithm has been used for the synthesis of FBG-based bandpass filters using the non-uniform FBG model based on the cascade of serially-connected *uniform* sub-gratings (Zheng et al., 2005).

4.2 Hybrid Tabu algorithm for optimization of non-uniform fiber Bragg gratings

The global SCTS algorithm (Section 4.1) generally cannot produce an optimum solution of the synthesized FBG, which may make practical realization difficult. Unfortunately, local optimization methods such as the Quasi-Newton method also cannot yield an optimum solution of the FBG synthesis problem due to the multimodal and ill-conditioned character of the nonlinear objective function. To improve the convergence of the global optimization algorithm, a hybrid algorithm combining the global optimization algorithm and the local optimization algorithm is a better approach for solving the nonlinear objective functions. The motivation of this section is to thus present a hybrid Tabu algorithm for the synthesis and fabrication of FBG-based bandpass and linear phase filters (Ngo, Zheng, Ng, Tjin & Binh, 2007). The hybrid Tabu algorithm (see Fig. 10) is a two-tier search that employs the global SCTS algorithm (Section 4.1) and a local optimization algorithm (i.e., the Quasi-Newton method which has high efficiency in solving multimodal nonlinear optimization problems (Shanno, 1970)).

In Fig. 10, the first step of the global SCTS algorithm produces a "promising" grating structure, which can either consist of the cascade of serially-connected *uniform* sub-gratings (Section 2.2.1) or *apodized* sub-gratings (Section 2.2.2), with the optimized structural parameters (δl_j, $\Delta n_{\mathrm{ac},j}$, f_j, Λ_j and $\Delta n_{\mathrm{dc},j}$). In this first step, the length of the "promising" index modulation profile is divided into N equally-spaced points, that is, $\Delta n_{\mathrm{ac},j}$ $(j=1,2,\cdots N)$ is sampled at the discrete point z_j, where $\Delta n_{\mathrm{ac},j}$ can be assumed to be constant for each sub-grating section. The index modulation profile, $\Delta n_{\mathrm{ac},j}$ $(j=1,2,\cdots N)$, of this "promising" grating structure is further optimized by the local Quasi-Newton algorithm. In this second step, the length of the "promising" index modulation profile is

divided into M equally-spaced points, that is, $\Delta n_{\mathrm{ac},j}$ $\left(j = 1, 2, \cdots M\right)$ is sampled at the discrete point z_j, where $\Delta n_{\mathrm{ac},j}$ can be assumed to be constant for each sub-grating section. The TMM method described in Section 2 can be used to calculate the frequency responses of the designed FBG-based filters. To demonstrate its effectiveness, the hybrid Tabu algorithm is applied to the design and fabrication of FBG-based bandpass filters and the design of FBG-based linear phase filters, and this is described below.

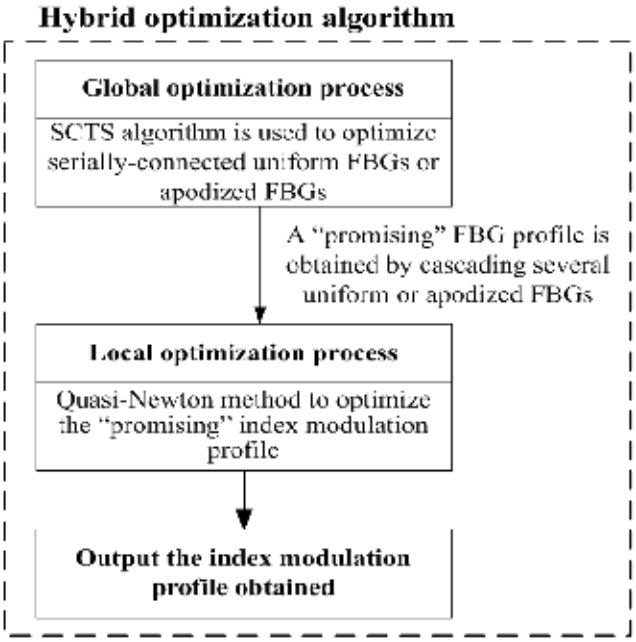


Fig. 10. Schematic diagram of the hybrid Tabu algorithm for use in the optimization of FBGs.

4.3 Design and fabrication of FBG-based bandpass filters

Bandpass optical filters are attractive for performing many signal processing and filtering functions in optical sensors and optical communications. This section describes the use of the hybrid Tabu algorithm described in Section 4.2 for the synthesis and fabrication of bandpass optical filters. As only the magnitude response (but not the phase response) of the bandpass filter is to be synthesized, the FBG model based on the cascade of serially-connected *uniform* (instead of *apodized*) sub-gratings (section 2.2.1) is employed here. The magnitude response of the FBG-based bandpass filter to be synthesized is a function of the index modulation profile, $\overrightarrow{\mathrm{Var}}$, and is given by

$$R_j\left(\overrightarrow{\mathrm{Var}}\right); \quad j \in \text{ wavelength window} \tag{30}$$

where $\overrightarrow{\mathrm{Var}} = \left[\Delta n_{\mathrm{ac},1},\ \Delta n_{\mathrm{ac},2},\ \cdots,\ \Delta n_{\mathrm{ac},M}\right]$. The target magnitude response of the bandpass filter with a bandwidth of 25 GHz (or 0.2 nm in the 1550 nm wavelength window) is given by

$$R_{\text{target},j} = \begin{cases} 1; & 1549.9 \text{ nm} \le \lambda_j \le 1550.1 \text{ nm} \\ 0; & \lambda_j < 1549.9 \text{ nm and } \lambda_j > 1550.1 \text{ nm} \end{cases} \tag{31}$$

The step size or wavelength resolution of 0.01 nm is used in the optimization. The objective function to be optimized is a measure of the error between the calculated magnitude response of the optimized FBG (Eq. (30)) and the target magnitude response (Eq. (31)) and is given by

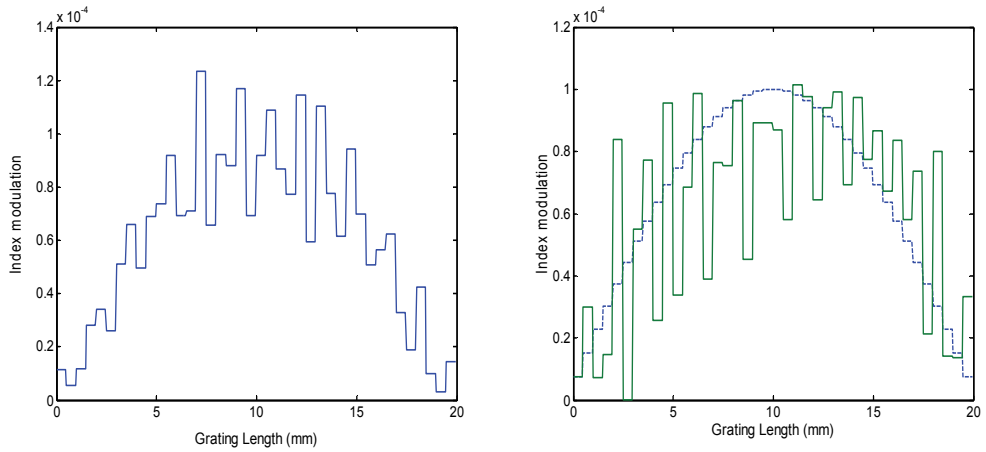
$$\text{error}(\overrightarrow{\Delta n_{\text{ac}}}) = \sum_{\substack{j \in \text{wavelength} \\ \text{window}}} W_j^R \times \sqrt{\left| R_j\left(\overrightarrow{\text{Var}}\right) - R_{\text{target},j} \right|} \tag{32}$$

where $R_j\left(\overrightarrow{\text{Var}}\right)$ is the reflectivity of the FBG's magnitude response at the j^{th} wavelength (λ_j) which is computed using the hybrid Tabu algorithm and Eq. (14) and $R_{\text{target},j}$ is the reflectivity of the target magnitude response at the j^{th} wavelength which is defined in Eq. (31). The weight parameter at the j^{th} wavelength is given by

$$W_j^R = \begin{cases} 10; & \lambda_j \in \text{stopband} \\ 1; & \lambda_j \in \text{passband} \end{cases} \tag{33}$$

where the weight values in the stopband are chosen to be 10 (ten) times more than those in the passpand in order to achieve effective suppression of the sidelobes. An additional constraint can be imposed on $\overrightarrow{\Delta n_{\text{ac}}} = \left[\Delta n_{\text{ac},1}, \Delta n_{\text{ac},2}, \cdots, \Delta n_{\text{ac},M}\right]$ by setting appropriate values of $\Delta n_{\text{ac},1}$ and $\Delta n_{\text{ac},M}$ to suit the condition(s) of a particular fabrication system. In this design, $M = 40$ and $\overrightarrow{\Delta n_{\text{ac}}} = \left[0,\ 0.0002\right]$ are chosen, and the chosen parameter values of the SCTS algorithm are listed in Table 1. The beam size of our frequency-doubled argon laser system is ~0.5 mm and thus the sub-grating length, δl_j, (which must be at least equal to the beam size) is chosen to be ~0.5 mm to meet the fabrication requirement. The hybrid Tabu algorithm is to find the global optimum of Eq. (32). The optimized bandpass filter using the hybrid Tabu algorithm has 40 sub-grating sections and thus a total grating length of $L_g = 20$ mm. Figure 11(a) shows the optimized index modulation profile, $\overrightarrow{\Delta n_{\text{ac}}}$, of the FBG-based bandpass filter based on the hybrid Tabu algorithm. For comparison, the solid line of Fig. 11(b) shows the index modulation profile obtained from the first stage of the hybrid Tabu method (i.e., the SCTS process, see Fig. 10), and this profile is used as the "promising" structure in the second stage of the hybrid method. The dashed line of Fig. 11(b) shows the index modulation profile of a standard sine-apodized FBG-based bandpass filter (without using optimization) (Kashyap, 1999). The corresponding reflection magnitude responses of these three index modulation profiles are shown in Fig. 12. The SCTS-optimized bandpass filter outperforms the standard apodized bandpass filter (without using optimization) in terms of sharper roll-offs and larger sidelobe suppression levels, demonstrating the

advantage of employing optimization. The hybrid-optimized bandpass filter using the hybrid Tabu algorithm performs best because the roll-offs on the edges of the bandpass response are steepest and the sidelobes have the greatest suppression levels of less than –30 dB. The hybrid-optimized filter with results shown in Fig. 11(a) and Fig. 12 can be fabricated using the fabrication method reported by (Asseh, Storoy, Sahlgren, Sandgren & Stubbe, 1997; Ngo, Zheng, Ng, Tjin & Binh, 2007), and this is described as follows. The 40 sub-gratings in a 20-mm long grating were exposed in sequence using UV pulses. Each sub-grating has a few hundred periods. The index-modulation depth of each sub-grating can be tuned to the designed value (as shown in Fig. 11(a)) by adjusting the offset of the fiber dithering away from the phase mask. That is, if the offset of the fiber dithering is half of the sub-grating period, the index modulation will be completely averaged out (i.e., no index modulation). However, if there is no offset of the fiber dithering from the phase mask, the index modulation value will be largest. In Fig. 13, the measured reflection magnitude response of the fabricated 20-mm long hybrid-optimized FBG-based bandpass filter using this fabrication method (the solid line) has steeper roll-offs than those of the fabricated 20-mm long standard (or non-optimized) uniform FBG-based bandpass filter (the dashed line). The figure shows that the fabricated filter has a sidelobe suppression level of –20 dB which is sufficient for many practical applications. This suppression level is greater than the designed –30 dB level (see Fig. 12) due probably to fabrication errors such as the positioning error of the translation stage, the fluctuation of the UV laser power and tiny dirty spots on the phase mask.



(a) (b)

Fig. 11. (a) The optimized index modulation profile, $\overrightarrow{\Delta n_{ac}}$, of the FBG-based bandpass filter based on the hybrid Tabu algorithm. (b) The solid line shows the index modulation profile obtained from the first stage of the hybrid Tabu method. The dashed line shows the index modulation profile of a standard sine-apodized FBG-based bandpass filter (without using optimization).

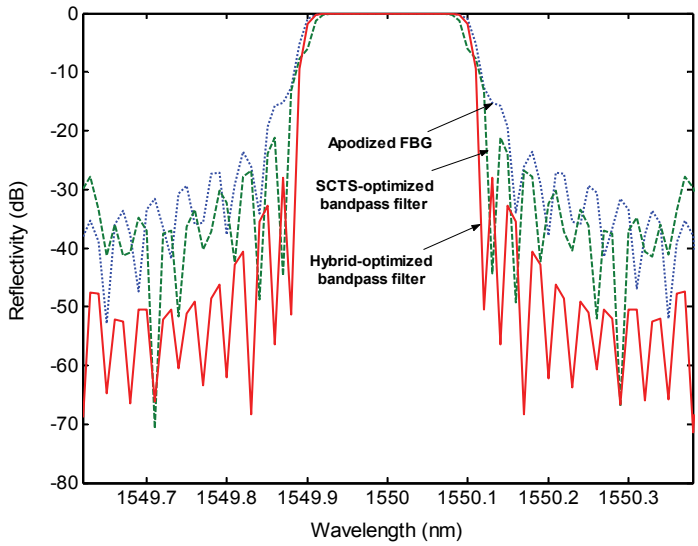


Fig. 12. Simulation results of the reflection magnitude responses of the three index modulation profiles shown in Fig. 11.

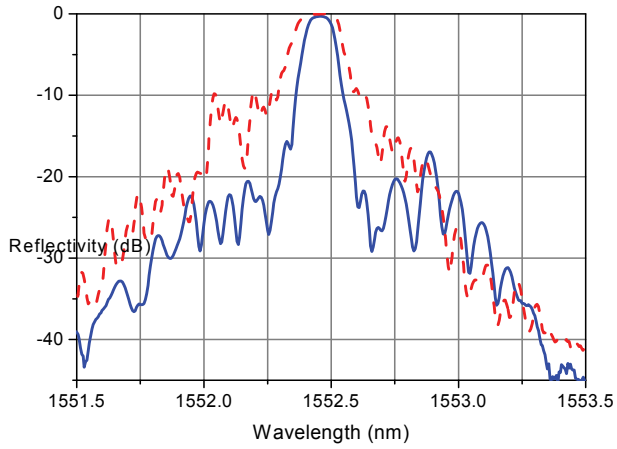


Fig. 13. Measured reflection magnitude responses of a 20-mm long hybrid-optimized FBG-based bandpass filter (solid line) and a standard 20-mm long uniform FBG-based bandpass filter (without using optimization) (dashed line).

4.4 Design of FBG-based linear phase filters

Linear phase optical filters with near-squared magnitude responses are attractive for signal processing and filtering in optical sensors and optical communications systems while maintaining the essential features of the signals. One important application of such a filter is a high-speed (> 10 Gbit/s) add/drop multiplexer that can add or drop a particular signal wavelength without introducing distortion to the signal. To further demonstrate the

effectiveness of the hybrid Tabu method, this section presents the design of three FBG-based linear phase filters with different grating lengths and with 50 GHz bandwidth each (i.e., 0.4 nm in the 1550-nm wavelength window). Both the magnitude and phase responses of the linear phase filters are to be synthesized using the hybrid Tabu algorithm. Two stages are involved in the hybrid Tabu algorithm and they are described below. The first stage of the hybrid Tabu algorithm employs the non-uniform FBG model based on the cascade of serially-connected *apodized* sub-gratings (Section 2.2.2). The second stage of the hybrid Tabu algorithm employs the non-uniform FBG model based on the cascade of serially-connected *uniform* sub-gratings (Section 2.2.1) instead of *apodized* sub-gratings (Section 2.2.2) because it is much simpler to fabricate the former than the later.

First stage of the hybrid Tabu algorithm: The first stage of the hybrid Tabu algorithm is the global SCTS algorithm (see Fig. 10). The desired magnitude response of a FBG-based linear phase filter with a bandwidth of 0.4 nm (or 50 GHz in the 1550 nm wavelength window) to be synthesized is a function of the index modulation profile, $\overrightarrow{\text{Var}}$, and is given by

$$R_j\left(\overrightarrow{\text{Var}}\right); \quad j \in \text{ wavelength window} \tag{34}$$

The row vector $\overrightarrow{\text{Var}}$ is given by

$$\overrightarrow{\text{Var}} = \left[\, \Delta n_{\text{ac},\,1}^{\max}, \cdots, \Delta n_{\text{ac},\,N}^{\max},\ \delta l_1, \cdots, \delta l_N \,\right] \tag{35}$$

where $\Delta n_{\text{ac},j}^{\max}$ and δl_j are the maximum amplitude of the index modulation and the sub-grating length of the jth sub-grating, respectively, that are to be optimized. The design objective is to find $\overrightarrow{\text{Var}}$ that produces the optimized magnitude and linear phase responses as close as possible to the target ones. The parameters that are not optimized are Λ_j and $\Delta n_{\text{dc},j}(z) = \Delta n_{\text{dc},j}$ (and hence $\bar{n}_{\text{eff},j}$ according to Eq. (2)), which are the period and the "dc" index change, respectively, and they are fixed or constant values for ease of fabrication. The apodization function used is the quadratic-sine apodization profile which is defined in Eq. (21) as $f_j = \sin^2\left(\pi z / \delta l_j\right)$. Compared to other apodization profiles as defined in Eqs. (18)–(20), the quadratic-sine apodization profile can provide a larger sidelobe suppression level and a higher reflectivity in the passband of the magnitude response. The step size or wavelength resolution of 0.01 nm is used in the optimization. The target magnitude response of a linear phase filter (with 0.4 nm or 50 GHz bandwidth in the 1550 nm wavelength window) is given by

$$R_{\text{target},j} = \begin{cases} 1; & 1549.8 \text{ nm} \le \lambda_j \le 1550.2 \text{ nm (passband)} \\ 0; & \lambda_j < 1549.8 \text{ nm and } \lambda_j > 1550.2 \text{ nm (stopband)} \end{cases} \tag{36}$$

The objective function to be optimized is a measure of the error between the optimized magnitude response of the FBG (Eq. (30)) and the target magnitude response (Eq. (36)) and is given by

$$\text{error}(\overrightarrow{\text{Var}}) = \sum_{j \in \text{window}} W_j^R \times \sqrt{\left| R_j(\overrightarrow{\text{Var}}) - R_{\text{target},j} \right|} + b \times \sum_{k \in \text{passband}} \left| D_k(\overrightarrow{\text{Var}}) - D_{\text{target},k} \right| \tag{37}$$

where $R_j(\overrightarrow{\text{Var}})$ is the reflectivity of the FBG's magnitude response at the j^{th} wavelength (λ_j) which is calculated based on the hybrid Tabu algorithm and Eq. (14). In the stopband, suppression of sidelobes in wavelength regions close to the center wavelength is more critical than in other wavelength regions. To achieve this and to increase the optimization efficiency, the weight parameter of the j^{th} wavelength, W_j^R, for the reflectivity of the magnitude response is given by

$$W_j^R = \begin{cases} \dfrac{\lambda_0}{\left|\lambda_0 - \lambda_j\right|}; & \lambda_j \in \text{stopband} \\ \varepsilon; & \lambda_j \in \text{passband} \end{cases} \tag{38}$$

where λ_0 is the center wavelength. The numerical value of ε was chosen by means of trials and errors by carrying out many simulations, and it was found that $\varepsilon = 1$ gave promising results that meet the specifications of the filter designs. However, in general, it is possible that other values of ε could give satisfying optimization results. $D_k(\overrightarrow{\text{Var}})$ is the dispersion at the k^{th} wavelength which is calculated using the hybrid Tabu algorithm and Eq. (16). The target dispersion at the k^{th} wavelength of a linear phase filter is given by

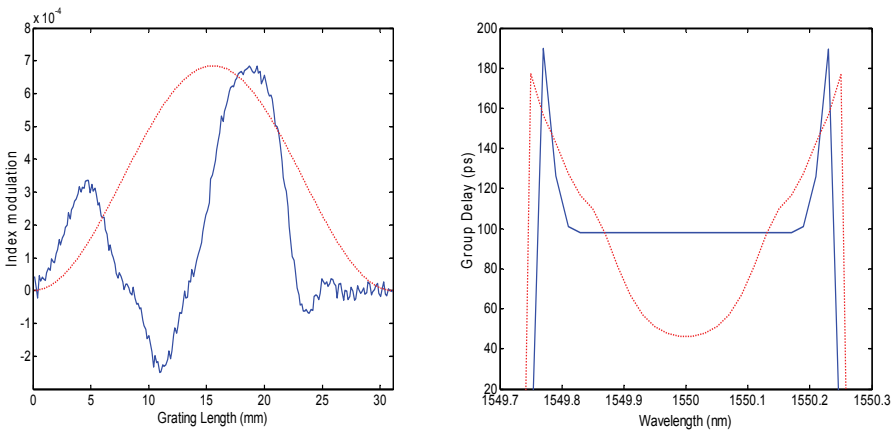
$$D_{\text{target},\,k} = 0; \quad \lambda_k \in \text{passband}. \tag{39}$$

The adjustable parameter b is used to dynamically balance the error between the reflectivity of the magnitude response and the dispersion response. The first stage of the hybrid Tabu algorithm (i.e., the SCTS algorithm) is to find the global optimum of Eq. (37). Additional constraints (e.g., Λ_j) can be included in $\overrightarrow{\text{Var}}$ in (Eq. (35) to suit the fabrication condition(s) or limitation(s) of a particular fabrication system to ensure that the designed filter can be practically realized. From Eq. (35), the length of $\overrightarrow{\text{Var}}$ for N cascaded serially-connected *apodized* sub-gratings is $2 \times N$ because $\overrightarrow{\text{Var}}$ has two variables; thus the filter synthesis problem is an optimization problem with $2 \times N$ variables. Satisfying results can be achieved with $N = 5$ while minimizing the computation time. From Eq. (35), elements from the $(N + 1)th$ to the $(2 \times N)th$ of the vector $\overrightarrow{\text{Var}}$ describe the lengths of the *apodized* sub-gratings. By setting appropriate boundary values for these elements, the total grating length L_{g} can be controlled in the optimization to make fabrication feasible.

Second stage of the hybrid Tabu algorithm: After a "promising" index modulation profile (see Eq. (35)) is obtained from the first stage of the hybrid Tabu process as described above, it is set as the initial solution for the second stage of the hybrid Tabu algorithm. In the second stage of the hybrid method (i.e., the local Quasi-Newton method), the index modulation profile is divided into M sections and hence M variables in the optimization. The M sections are equally spaced. The index modulation of each section is assumed to be constant. The non-uniform FBG model employed in this second stage is based on the cascade of serially-connected *uniform* sub-gratings (Section 2.2.1) instead of *apodized* sub-gratings (Section 2.2.2) because it is much easier to fabricate the former than the later. Thus the error function for the index modulation, $\overrightarrow{\Delta n_{\text{ac}}} = \left[\Delta n_{\text{ac},\,1},\ \Delta n_{\text{ac},\,2},\ \cdots,\ \Delta n_{\text{ac},\,M}\right]$, to be optimized can be defined as

$$\text{error}\left(\overrightarrow{\Delta n_{\text{ac}}}\right) = \sum_{j \in \text{window}} W_j^R \times \sqrt{\left| R_j\left(\overrightarrow{\Delta n_{\text{ac}}}\right) - R_{\text{target},j} \right|} + b \times \sum_{k \in \text{passband}} \left| D_k\left(\overrightarrow{\Delta n_{\text{ac}}}\right) - D_{\text{target},k} \right| \tag{40}$$

where $R_j\left(\overrightarrow{\Delta n_{\text{ac}}}\right)$ is the reflectivity of the FBG's magnitude response at the j^{th} wavelength (λ_j) which is calculated based on the hybrid Tabu algorithm and Eq. (14), $D_k\left(\overrightarrow{\Delta n_{\text{ac}}}\right)$ is the dispersion at the k^{th} wavelength which is calculated using the hybrid Tabu algorithm and Eq. (16), W_j^R is defined in Eq. (38), $R_{\text{target},j}$ is defined in Eq. (36), and $D_{\text{target},k}$ is defined in Eq. (39). The adjustable parameter b is used to dynamically balance the error between the reflectivity of the magnitude response and the dispersion response. The local Quasi-Newton method is employed to find the optimum of Eq. (40). The optimization results of the design of an FBG-based linear phase filter with a grating length of 31.1 mm are presented next. The solid curve of Fig. 14(a) shows the index modulation profile of an optimized FBG-based linear phase filter with a length of 31.1 mm using the hybrid Tabu algorithm; the corresponding group-delay response is shown as the solid curve in Fig. 14(b) and the corresponding reflection magnitude response is shown as the solid curve in Fig. 15. The index modulation profile (dotted curve of Fig. 14(a)), the group-delay response (dotted curve of Fig. 14(b)) and the reflection magnitude response (dotted curve of Fig. 15) of a 31.1-mm long quadratic-sine apodized FBG-based linear phase filter obtained from the standard design method (without using optimization) (Kashyap, 1999) are also shown for comparison purpose. In Fig. 15, the almost-squared reflection magnitude response of the optimized FBG-based linear phase filter has a good sidelobe suppression level of < –45 dB (which is acceptable for many practical applications), while the almost-squared reflection magnitude response of the unoptimized FBG-based linear phase filter has a very good sidelobe suppression level of < –55 dB. Fig. 14(b) shows that the in-band group delay ripple of the optimized linear phase filter is < 0.3 ps (the solid curve), which is a lot flatter than that of the unoptimized linear phase filter (the dotted curve). It can thus be concluded that the



(a) (b)

Fig. 14. Index modulation profile (solid curve) (a) and the corresponding group-delay response (solid curve) (b) of an optimized 31.1-mm long FBG-based linear phase filter using the hybrid Tabu algorithm. Index modulation profile (dotted curve) (a) and the corresponding group-delay response (dotted curve) (b) of a 31.1-mm long quadratic-sine apodized FBG using the standard design method (without using optimization).

optimized FBG-based linear phase filter using the hybrid Tabu algorithm has linear phase characteristics and near-ideal squared reflection magnitude response which meet the design specifications. Table 4 also shows the performances of other optimized linear phase filters with shorter grating lengths of 17.1 mm and 25.8 mm. The designed filter with the longest grating length of 31.1 mm has the smallest sidelobe level of –45 dB while the designed filter with the shortest grating length of 17.1 mm has the largest sidelobe of –35 dB. However, the group delay ripple increases with the grating length of the designed filter. The designed filter with the longest grating length of 31.1 mm has the largest in-band group delay ripple of 0.3 ps while the designed filter with the shortest grating length of 17.1 mm has the smallest group delay ripple of 0.1 ps. Thus a long grating length will yield a large sidelobe level but at the expense of a large group delay ripple. We can thus conclude that there is a trade-off between the maximum sidelobe value and the maximum in-band group delay ripple for a particular grating length.

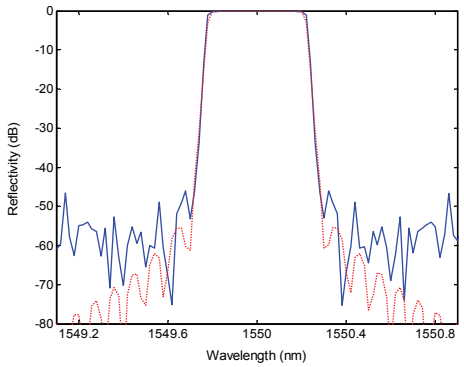


Fig. 15. The solid curve is the corresponding magnitude response of the index modulation profile (the solid curve of Fig. 14(a)) of a 31.1-mm long optimized linear phase filter obtained from the hybrid Tabu algorithm. The dotted curve is the corresponding magnitude response of the index modulation profile (the dotted curve of Fig. 14(a)) of a 31.1-mm long quadratic-sine apodized FBG obtained from the standard design method (without using optimization).

Grating length (mm)	Maximum sidelobe value (dB)	Maximum in-band group delay ripple (ps)
17.1	–35	0.1
25.8	–37	0.2
31.1	–45	0.3

Table 4. Performance comparison of three optimized FBG-based linear phase filters with different grating lengths using the hybrid Tabu algorithm.

Two main issues involved in the fabrication of the optimized index modulation (i.e., the "ac" index change) profile shown in the solid curve of Fig. 14(a) are described as follows. First, the fabrication process must be able to realize such a relatively complex index modulation profile along the grating length while maintaining an unperturbed "dc" index change along the grating length. The fabrication process required to realize this optimized index modulation profile is similar to that used in the fabrication of the optimized apodized index modulation profile shown in Fig. 11(a). This is because the fabrication of these two types of profiles is to produce some desired index modulation values corresponding to the grating positions while maintaining an unperturbed "dc" index change. To meet this first

requirement, a well-developed technique of fiber dithering (which involves dithering of the fiber during the UV exposure) can be used (Asseh et al., 1997). In this method, the first step is to divide the whole grating into a number of sub-gratings. Each sub-grating inscribed by the fiber dithering will cause the index modulation depth to be averaged out. That is, if the offset of the fiber dithering is half of the grating period, the index modulation will be completely averaged out (i.e., no index modulation). Now if there is no offset of the fiber dithering from the phase mask, the index modulation will be maximum. Note that, to maintain an unperturbed 'dc' index change along the grating length, all the sub-gratings must be exposed with the same total number of UV pulses (i.e., the exposure energy levels of all the sub-gratings are the same). Second, the fabrication process must be able to introduce some π phase shifts (i.e., negative index modulation values) along the grating length (see solid curve of Fig. 14(a)). To meet this second requirement, the fiber should be moved relative to the phase mask (which is also a well-developed method), and this would enable the insertion of π phase shifts into the gratings during the fabrication process (Asseh et al., 1997; Cole et al., 1995).

5. Conclusion

A hybrid Tabu algorithm has been presented for the synthesis and fabrication of FBG-based filters with specified frequency responses. The hybrid Tabu algorithm is a two-tier search that employs the global staged continuous Tabu search (SCTS) algorithm and a local optimization algorithm (i.e., the Quasi-Newton method). The first step of the hybrid Tabu algorithm (which employs the global SCTS algorithm) produces a "promising" grating structure which is then further optimized by the second stage of the hybrid Tabu algorithm (which employs the Quasi-Newton algorithm) to yield the final optimized solution or grating structure. In the hybrid Tabu algorithm, the synthesis problem of an FBG is formulated as an objective function which measures the error between the optimized reflection frequency response of an FBG based on the hybrid Tabu algorithm and the target frequency response, and the optimized solution or grating structure has the smallest error. The FBG-based filters are based on non-uniform FBGs with a wide variety of frequency responses. The transfer matrix method (TMM) has been employed to model a non-uniform FBG because it allows the magnitude and phase responses of a non-uniform FBG to be obtained with reasonably high accuracy. In TMM, a non-uniform FBG can be modeled using either the cascade of serially-connected *uniform* sub-gratings (this model is sufficient for obtaining the magnitude response when the phase response is not important) or the cascade of serially-connected *apodized* sub-gratings (this model must be used when both the magnitude and phase responses are important). The effectiveness of the hybrid Tabu algorithm has been demonstrated through the design and fabrication of 25-GHz-bandwidth FBG-based bandpass filters with near-squared bandpass responses and the design of 50-GHz-bandwidth FBG-based linear phase filters with near-squared bandpass responses and flat group delay responses, which have potential application in optical communication systems operating in the 1550 nm wavelength window. For future work, the presented hybrid Tabu algorithm can be further developed into a powerful toolbox that can be efficiently and reliably used for the synthesis and fabrication of a wide variety of FBG-based filters with specified complex frequency responses to meet the increasing demand of sophisticated signal processing and filtering functions of the next generation of optical sensors and optical communication systems. Furthermore, the powerful design toolbox must be powerful enough for use in the diagnosis or characterization during and after fabrication of an FBG.

6. References

Andre, J.; Siarry, P. & Dognon, T. (2000). An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization. *Advances in Engineering Software*, Vol. 32, No. 1, (December 2000), pp. 49–60. ISSN 0965-9978.

Asseh, A; Storoy, H; Sahlgren, B. E.; Sandgren, S. & Stubbe R. (1997). A writing technique for long fiber Bragg gratings with complex reflectivity profiles. *Journal of Lightwave Technology*, Vol. 15, No. 8, (August 1997), pp. 1419–1424. ISSN 0733-8724.

Baskar, S.; Zheng, R. T.; Alphones, A.; Ngo, N. Q. & Suganthan, P. N. (2005a). Particle swarm optimization for the design of low-dispersion fiber Bragg gratings. *IEEE Photonics Technology Letters*, Vol. 17, No. 3, (March 2005), pp. 615−617. ISSN 1041-1135.

Baskar, S.; Alphones, A.; Suganthan, P. N.; Ngo, N. Q. & Zheng, R. T. (2005b). Design of optimal length low-dispersion FBG filter using covariance matrix adapted evolution. *IEEE Photonics Technology Letters*, Vol. 17, No. 10, (October 2005), pp. 2119–2121. ISSN 1041-1135.

Baskar, S.; Suganthan, P. N.; Ngo, N. Q.; Alphones, A. & Zheng, R. T. (2006). Design of triangular FBG filter for sensor applications using covariance matrix adapted evolution algorithm. *Optics Communications*, Vol. 260, No. 2, (April 2006), pp. 716−722. ISSN 0030-4018.

Caucheteur, C.; Lhomme, F.; Chah, K.; Blondel, M. & Megret, P. (2004). Fiber Bragg grating sensor demodulation technique by synthesis of grating parameters from its reflection spectrum. *Optics Communications*, Vol. 240, No. 4–6, (October, 2004), pp. 329–336. ISSN 0030-4018.

Chelouah, R. & Siarry, P. (2000). Tabu search applied to global optimization. European Journal of Operational Research, Vol. 123, No. 2, (June 2000), pp. 256–270. ISSN 0377-2217

Cheng, H.-C. & Lo, Y.-L. (2004). Arbitrary strain distribution measurement using a genetic algorithm approach for two fiber Bragg grating intensity spectra. *Optics Communications*, Vol. 239, No. 4–6, (September 2004), pp. 323–332. ISSN 0030-4018.

Cole, M. J.; Loh, W. H.; Laming, R. I.; Zervas, M. N. & Barcelos, S. (1995). Moving fibre/phase mask-scanning beam technique for enhanced flexibility in producing fiber gratings with uniform phase mask. *Electronics Letters,* Vol. 31, No. 17, (August 1995), pp. 1488–1450. ISSN 0013-5194.

Dong, P.; Azana, J. & Kirk, A. G. (2003). Synthesis of fiber Bragg grating parameters from reflectivity by means of a simulated annealing algorithm. *Optics Communications*, Vol. 228, No. 4–6, (December 2003), pp. 303–308. ISSN 0030-4018.

Erdogan, T. (1997). Fiber grating spectra. *Journal of Lightwave Technology*, Vol. 15, No. 8, (August 1997), pp. 1277–1294. ISSN 0733-8724.

Feced, R.; Zervas, M. N. & Muriel, M. A. (1999). An efficient inverse scattering algorithm for the design of nonuniform fiber Bragg gratings. *IEEE Journal in Quantum Electronics*, Vol. 35, No. 8, (August 1999), pp. 1105–1115. ISSN 0018-9197.

Gill, A.; Peters, K. & Studer, M. (2004). Genetic algorithm for the reconstruction of Bragg grating sensor strain profiles. *Measurement Science and Technology*, Vol. 15, No. 9, (September 2004), pp. 1877–1884. ISSN 0957-0233.

Glover, F. & Laguna, M. (1998). *Tabu search*, Kluwer Academic Publishers, Boston. ISBN 0792381874.

Hill, K. O. & Meltz, G. (1997). Fiber Bragg grating technology: fundamentals and overview. *Journal of Lightwave Technology*, Vol. 15, No. 8, (August 1997), pp. 1263–1276. ISSN 0733-8724.

Ho, S. L.; Yang, S. Y.; Ni, G. Z. & Wong, H. C. (2001). An improved tabu search for the global optimizations of electromagnetic devices. *IEEE Transactions on Magnetics*, Vol. 37, No. 5, (September 2001), pp. 3570–3574. ISSN 0018-9464.

Kashyap, R. (1999). *Fiber Bragg gratings*, Academic Press, ISBN 0-12-400560-8, San Diego.

Machado, J. M.; Yang, S.; Ho, S. L. & Ni, P. (2001). A common tabu search algorithm for the global optimization of engineering problems. *Computer Methods in Applied Mechanics and Engineering*, Vol. 190, No. 26–27, (March 2001), pp. 3501–3510. ISSN 0045-7825.

Manos, S. & Poladian, L. (2005). Multi-objective and constrained design of fibre Bragg gratings using evolutionary algorithms. *Optics Express*, Vol. 13, No. 19, (September 2005), pp. 7350–7364. ISSN #8059.

Ngo, N. Q.; Zheng, R. T.; Tjin, S. C. & Li, S. Y. (2004). Tabu search synthesis of cascaded fiber Bragg gratings for linear phase filters. *Optics Communications*, Vol. 241, No. 1-3, (November, 2004), pp. 79−85. ISSN 0030-4018.

Ngo, N. Q.; Zheng, R. T.; Ng, J. H.; Tjin, S. C. & Binh, L. N. (2007). Optimization of fiber Bragg gratings using a hybrid optimization algorithm. *Journal of Lightwave Technology*, Vol. 25, No. 3, (March 2007), pp. 799–802. ISSN 0733-8724.

Othonos, A. & Kalli, K. (1999). *Fiber Bragg gratings: fundamentals and applications in telecommunications and sensing*, Artech House, ISBN 0-89006-344-3, Boston.

Plougmann, N. & Kristensen, M. (2004). Efficient iterative technique for designing Bragg gratings. *Optics Letters*, Vol. 29, No. 1, (January 2004), pp. 23–25. ISSN 0146-9592.

Poladian, L. (2000). Simple grating synthesis algorithm. *Optics Letters,* Vol. 25, No. 11, (June 2000), pp. 787–789. ISSN 0146-9592.

Rosenthal, A. & Horowitz, M. (2003). Inverse scattering algorithm for reconstructing strongly reflecting fiber Bragg gratings. *IEEE Journal in Quantum Electronics*, Vol. 39, No. 8, (August 2003), pp. 1018–1026. ISSN 0018-9197.

Shanno, D. F. (1970). Conditioning of Quasi-Newton methods for function minimization. *Mathematics of Computation*, Vol. 24, no. 111, (July 1970), pp. 647–656. ISSN 0025-5718.

Siarry, P. & Berthiau, G. (1997). Fitting of tabu search to optimize functions of continuous variables. *International Journal for Numerical Methods in Engineering*, Vol. 40, No. 13, (July 1997), pp. 2449–2457. ISSN 0029-5981.

Skaar, J. & Risvik, K. M. (1998). A genetic algorithm for the inverse problem in synthesis of fiber gratings. *Journal of Lightwave Technology*, Vol. 16, No. 10, (October 1998), pp. 1928–1932. ISSN 0733-8724.

Skaar, J.; Wang, L. & Erdogan, T. (2001). On the synthesis of fiber Bragg gratings by layer peeling. *IEEE Journal in Quantum Electronics*, Vol. 37, No. 2, (February 2001), pp. 165–173. ISSN 0018-9197.

Zheng, R. T.; Ngo, N. Q.; Binh, L. N. & Tjin, S. C. (2004). Two-stage hybrid optimization of fiber Bragg gratings for design of linear phase filters. *Journal of Optical Society of America A: Optics Image Science and Vision*, Vol. 21, No. 12, (December 2004), pp. 2399−2405. ISSN 1084-7529.

Zheng, R. T.; Ngo, N. Q.; Shum, P; Tjin, S. C. & Binh, L. N. (2005). A staged continuous tabu search algorithm for the global optimization and its applications to the design of fiber Bragg gratings. *Computational Optimization and Applications*, Vol. 30, No. 3, (March 2005), pp. 319−335. ISSN 09266003.

Appendix: List of test functions

- *F1 (1 variable)*: $f(x) = 2(x-0.75)^2 + \sin(5\pi x - 0.4\pi) - 0.125$, where $0 \le x \le 1$.
- *F3 (1 variable)*: $f(x) = -\sum_{i=1}^{5} i \sin\left[(i+1)x + i\right]$, where $-10 \le x \le 10$.
- *Branin (2 variables)*: $f(x,y) = a\left(y - bx^2 + cx - d\right)^2 + h(1-g)\cos(x) + h$,

  where $a = 1$, $b = \dfrac{5.1}{4\pi^2}$, $c = \dfrac{5}{\pi}$, $d = 6$, $h = 10$, $g = \dfrac{1}{8\pi}$.
- *Goldprice (2 variables)*: $f(x,y) = \left[1 + (x+y+1)^2\left(19 - 14x + 3x^2 - 14y + 6xy + 3y^2\right)\right] \times \left[30 + (2x-3y)^2\left(18 - 32x + 12x^2 + 48y - 36xy + 27y^2\right)\right]$,

  where $-2 \le x \le 2$, $-2 \le y \le 2$.
- *Shubert1 and Shubert2 (2 variables):*

$$f(x,\, y) = \left\{\sum_{j=1}^{5} j \cdot \cos\left[(j+1)x + j\right]\right\} \times \left\{\sum_{j=1}^{5} j \cdot \cos\left[(j+1)y + j\right]\right\} + \beta\left[(x + 1.42513)^2 + (y + 0.80032)^2\right]$$

  where $-10 \le x, y \le 10$, $\beta = 0.5$ for Shuber1, and $\beta = 1$ for Shuber2.
- *Shubert (2 variables)*: $f(x_1,\, x_2) = \left\{\sum_{j=1}^{5} j \cdot \cos\left[(j+1)x_1 + j\right]\right\} \times \left\{\sum_{j=1}^{5} j \cdot \cos\left[(j+1)x_2 + j\right]\right\}$

  where $-10 \le x_i \le 10$, $i = 1, 2$.
- *Hartmann1 ($H_{3,4}$) (3 variables):*

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right], \quad \text{where } 0 < x_j < 1 \text{ for } j = 1,2,3.$$

i	a_{ij}			c_i	p_{ij}		
	$j=1$	$j=2$	$j=3$		$j=1$	$j=2$	$j=3$
1	3.0	10.0	30.0	1.0	0.3689	0.1170	0.2673
2	0.1	10.0	35.0	1.2	0.4699	0.4387	0.7470
3	3.0	10.0	30.0	3.0	0.1091	0.8732	0.5547
4	0.1	10.0	35.0	3.2	0.0382	0.5743	0.8828

- *Brown1 (20 variables):*

$$f(x)=\left[\sum_{i\in J}\left(x_i-3\right)\right]^2+\sum_{i\in J}\left[10^{-3}(x_i-3)^2-(x_i-x_{i+1})+\exp\left[20(x_i-x_{i+1})\right]\right]$$

where $J=\{1,\ 3,\ \cdots,\ 19\}$, $-1\le x_i\le 4$ for $1\le i\le 20$, and $x=[x_1,\ \cdots,\ x_{20}]^T$.

- *Brown3 (20 variables)*: $f(x)=\sum_{i=1}^{19}\left[\left(x_i^2\right)^{\left(x_{i+1}^2+1\right)}+\left(x_{i+1}^2\right)^{\left(x_i^2+1\right)}\right]$

where $x=\left[x_1,\ \cdots,\ x_{20}\right]^T$ and $-1\le x_i\le 4$ for $1\le i\le 20$.

- *F5n (20 variables)*:

$$f(x)=\left(\frac{\pi}{20}\right)\times\left\{10\sin^2\left(\pi y_1\right)+\Sigma\left[\left(y_i-1\right)^2\times\left(1+10\sin^2\left(\pi y_{i+1}\right)\right)\right]+\left(y_{20}-1\right)^2\right\}$$

where $y_i=1+0.25(x_i-1)$, $x=[x_1,\cdots,x_{20}]^T$, $-10\le x_i\le 10$.

- *F10n (20 variables)*:

$$f(x)=\left(\frac{\pi}{20}\right)\times\left\{10\sin^2\left(\pi x_1\right)+\sum_{i=1}^{19}\left[\left(x_i-1\right)^2\times\left(1+10\sin^2\left(\pi x_{i+1}\right)\right)\right]+\left(x_{20}-1\right)^2\right\}$$

where $x=\left[x_1,\ x_2,\ \cdots,\ x_{20}\right]^T$ and $-10\le x_i\le 10$.

- *F15n (20 variables)*:

$$f(x)=\left(\frac{1}{10}\right)\left\{\sin^2\left(3\pi x_1\right)+\sum_{i=1}^{19}\left[\left(x_i-1\right)^2\left(1+\sin^2\left(3\pi x_{i+1}\right)\right)\right]+\left(\frac{1}{10}\right)\left(x_{20}-1\right)^2\left[1+\sin^2\left(2\pi x_{20}\right)\right]\right\}$$

where $x=\left[x_1,\ x_2,\ \cdots,\ x_{20}\right]^T$ and $-10\le x_i\le 10$.

8

Some New Results on Tabu Search Algorithm Applied to the Job-Shop Scheduling Problem

Chaoyong Zhang, Xinyu Shao, Yunqing Rao and Haobo Qiu
Key Laboratory of Digital Manufacturing Equipment & Technology, School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074 China

1. Introduction

The Job-Shop Scheduling Problem (JSSP) with the makespan criterion comes from the manufacturing industry and has excellent practical applications. The problem can be briefly described as follows. There are a set of jobs and a set of machines. Each job consists of a sequence of operations, and each of the operations uses one of the machines for a fixed duration. Each machine processes at most one operation at one time. Once the operation started, no preemption is permitted. A scheduling is an assignment of operations to time intervals on the machines. The objective of the problem is to find a schedule which minimizes the makespan (C_{max}), that is, the finish time of the last operation completed.
The JSSP is widely acknowledged as one of the most difficult NP-complete problems (Garey et al., 1976). This is illustrated by the fact that a relatively small instance with 10 jobs and 10 machines, proposed by Muth & Thompson (1963), remained unsolved for more than a quarter of a century, and until now no problems are solved to optimality for the 20×20 instances. Since it is an important practical problem, the JSSP has captured the interest of a significant number of researchers during the past three decades, and many optimization algorithms and approximation algorithms have been proposed. The optimization algorithms, which are primarily based on the B&B scheme (Carlier & Pinson, 1989; Brucker et al., 1994), have been successfully applied to solving small instances. However, they cannot accomplish optimal schedules in a reasonable time for instances larger than 250 operations with reached the limit. On the other hand, approximation algorithms, which include priority dispatch, shifting bottleneck approach, meta-heuristic methods and so on, provide a quite good alternative for the JSSP. Approximation algorithms were firstly developed on the basis of dispatching rules (Giffler & Thompson, 1960), which are very fast, but the quality of solutions that they provide usually leaves plenty of room for improvement. A more elaborate algorithm, which could produce considerably better approximations at a higher computational cost, is the shifting bottleneck approach proposed by Adams et al. (Adams et al., 1988). More recently, the meta-heuristic methods, such as tabu search (TS) (Taillard, 1994; Nowicki & Smutnicki, 1996), simulated annealing (SA) (Van Laarhoven et al., 1992), genetic algorithm (GA) (Croce et al., 1995), could provide the good solutions for a large scale problem and have captured the attention of many researchers. Moreover, most recent studies indicate that a single technique cannot solve this stubborn

problem. Much work has been directed on hybrid methods involving GA, TS, SA and SB techniques, as hybrid methods are able to provide high-quality solutions within reasonable computing times. The relevant surveys can be seen form Vaessens et al. (1996), Blażewicz et al. (1996) and Jain & Meeran (1999).

Within the class of meta-heuristic methods, Tabu search, initially proposed by Glover (Glover, 1989 1990; Glover & Laguna, 1997) and Hansen (Hansen, 1986), currently seems to be one of the most promising methods for the job shop scheduling problem with the makespan criterion. It uses a memory function to avoid being trapped at a local minimum. Tabu search was firstly applied to the JSSP by Taillard (1989), whose main contribution was the use of the neighborhood structure introduced by Van Laarhoven et al. (1992) and the presentation of a fast move estimation strategy. Furthermore, Taillard (1989) observed that this algorithm has a higher efficiency for rectangular instances. Since then, researchers have introduced numerous improvements to Taillard's original algorithm, and the most important contributions are made by a myriad of researchers among whom are Nowicki & Smutnicki (1996), Dell'Amico & Trubian (1993), Barnes & Chambers (1995) and Chambers & Barnes (1996). Among these individual tabu search methods, algorithm TSAB designed by Nowicki & Smutnicki (1996) introduces the real breakthrough in both efficiency and effectiveness for the JSSP. For example, it finds the optimal solution for the notorious instance FT10 within only 30 seconds on a now-dated personal computer. The *i*-TSAB technique of Nowicki & Smutnicki (2002), which is an extension of their earlier TSAB algorithm, represents the current state-of-the-art approximation algorithm for the JSSP and improves the majority of upper bounds of the unsolved instances.

Although tabu search has emerged as an effective algorithmic approach for the JSSP, it was initially designed to find the near-optimum solution of combinatorial optimization problems and no clean proof of convergence is known (Hanafi, 2000). Like many local searches the quality of the best solution found by tabu search approach depends on the initial solution and neighborhood structures. In this paper, two innovative approaches are proposed to overcome these problems for the JSSP. Firstly, a new neighborhood structure is proposed to solve the job shop scheduling problem by tabu search approach. Secondly, by reasonably combining the memory function (avoid cycling) of tabu search and the convergent characteristics of simulated annealing, we develop an efficient hybrid optimization algorithm. In this approach, simulated annealing is used to find the sufficient "good" solutions over the big valley so that tabu search can re-intensify searches from the promising solutions.

In the end, this algorithm is tested on the commonly standard benchmark set and compared with the other approaches. The computational results show that the proposed algorithm could reduce the influence of the initial solution and obtain the high-quality solutions within reasonable computing times. These have been confirmed by tests on a large number of benchmark problems. For example, some new upper bounds among the unsolved problems are found in a short time.

The remainder of this paper is organized as follows. Section 2 gives the representation of the job shop scheduling problem. In Section 3, the framework of the hybrid of tabu search and simulated annealing is provided. Section 4 presents the implementation of TSSA algorithm for the JSSP and the proposed neighborhood structure. In Section 5, we firstly present the comparison of the different neighborhood structures and comparison of move evaluation strategies respectively, and then give the computational and comparative results on the benchmark instances. Conclusion is presented in Section 6.

2. Representation of the JSSP

The job shop scheduling problem can be represented with a disjunctive graph (Balas, 1969). Let $J = \{1, 2, \ldots, n\}$ be the set of jobs, $M = \{1, 2, \ldots, m\}$ the set of machines. A disjunctive graph $G: = (V, A, E)$ is defined as follows: V is $\{0, 1, 2, \ldots, \tilde{n}\}$ the set of nodes representing all operations where 0 and $\tilde{n}$ represent the dummy start and finish operations, respectively. A is the set of conjunctive (directed) arcs connecting consecutive operations of the same job, and E is the set of disjunctive arcs connecting operations to be processed by the same machine k. More precisely, $E = \bigcup_{k=1}^{m} E_k$, where E_k is the subset of disjunctive pair-arcs corresponding to machine k; each disjunctive arc of E can be considered as a pair of oppositely directed arc. The length of an arc $(i, j) \in A$ is p_i that denotes the processing time. The length of each arc $(i, j) \in E$ is either p_i or p_j depending on its orientation. Let us consider an example of the three jobs and three machines given in Table 1. This problem can be represented by a disjunctive graph shown in Fig. 1.

Job	(Machine sequence, Processing time)		
J1	(1, 3)	(2, 2)	(3, 5)
J2	(1, 3)	(3, 5)	(2, 1)
J3	(2, 2)	(1, 5)	(3, 3)

Table 1. An example of three jobs and three machines

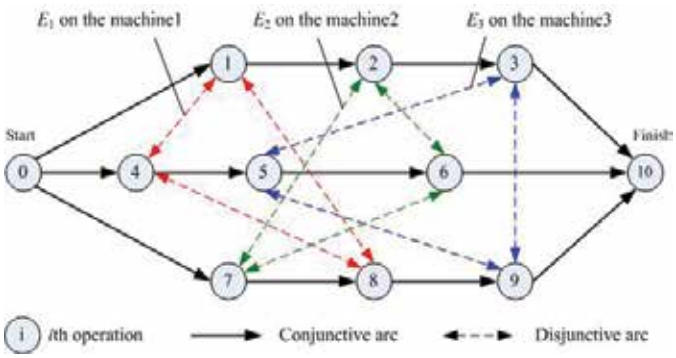


Fig. 1. The disjunctive graph of an instance with $n = 3$, $m = 3$, and $\tilde{n} = 10$

According to the Adams et al. (1988) method, the graph G can be decomposed into the direct sub-graph $D = (V, A)$, by removing disjunctive arcs, and into m cliques $G_k = (V_k, E_k)$, obtained from G by deleting both the conjunctive arcs and the dummy nodes 0 and $\tilde{n}$. A selection S_k in E_k contains exactly one directed arc between each pair of oppositely directed arcs in E_k. A selection is acyclic if it does not contain any directed cycle. Moreover, sequencing machine k means choosing an acyclic selection in E_k. A complete selection S consists of the union of selections S_k, one of each E_k, $k \in M$. A complete selection S, i.e., replacing the disjunctive arc set E with the conjunctive arc set S, gives rise to directed graph $D_s = (V, A \cup S)$; A complete selection S is acyclic if the digraph D_s is acyclic. An acyclic selection S defines a schedule, i.e., a feasible solution of problem. Fig. 2 represents a feasible solution for the disjunctive graph in Fig. 1. Furthermore, if $L(u, v)$ denotes the length of a longest path from u to v in D_s, then the makespan $L(0, \tilde{n})$ of the schedule is equal to the

length of a longest path in D_s. Therefore, in the language of disjunctive graphs, to solve the job shop scheduling problem is to find an acyclic complete selection $S \subset E$ that minimizes the length of the longest (critical) path in the directed graph D_s.

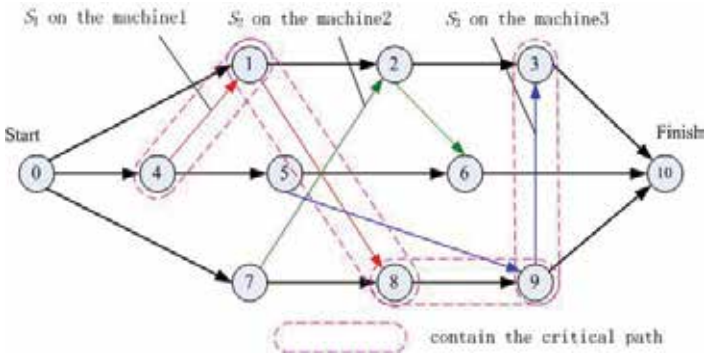


Fig. 2. A feasible solution for the disjunctive graph in Fig. 1

A key component of a feasible solution is the critical path, which is the longest route from start to finish in directed graph D_s= (V, $A \cup S$) and whose length represents the makespan C_{max}. Any operation on the critical path is called a critical operation. In Fig. 2 the length of the critical path is 19 and the critical path is {0, 4, 1, 8, 9, 3, 10}. It is also possible to decompose the critical path into a number of blocks. A block is a maximal sequence of adjacent critical operations that is processed on the same machine. In Fig. 2 the critical path is divided into two blocks, B_1 = {4, 1, 8} and B_2 = {9, 3}. Any operation, u, has two immediate predecessors and successors, with its job predecessor and successor denoted by $JP[u]$ and $JS[u]$ and its machine predecessor and successor denoted by $MP[u]$ and $MS[u]$. In other words, ($JP[u]$, u) and (u, $JS[u]$) are arcs of the conjunctive graph D_s, ($MP[u]$, u) and (u, $MS[u]$) (if they exist) are arcs of S.

In the JSSP, small perturbations are generally produced by re-ordering the sequence of operations on a critical path, and only through such re-ordering is it possible to produce a neighbor with a makespan better than that of the current solution.

3. The neighborhood structure

A neighborhood structure is a mechanism which can obtain a new set of neighbor solutions by applying a small perturbation to a given solution. Each neighbor solution is reached immediately from a given solution by a move (Glover & Laguna, 1997). Neighborhood structure is directly effective on the efficiency of tabu search algorithm. Therefore, unnecessary and infeasible moves must be eliminated if it is possible.

The most general neighborhood definition consists of swapping any adjacent pair of operations on the same machine. This neighborhood is quite large, and requires considerable effort to identify and evaluate the schedule that results from each possible move. For large problems the neighborhood contains more moves than can be examined and evaluated within a reasonable time. In addition, some of the moves can result in non feasible schedules. Consequently work has been devoted to the goal of reducing the size of this neighborhood and guaranteeing feasibility but without affecting solution quality (Jain et al., 2000).

The first successful neighborhood structure for the JSSP was introduced by van Laarhoven et al. (1992), and is often denoted by N1, see Fig. 3 (N1 is first named by Blazewicz et al (1996), and N2, N4, N5 and N6 are named in the same way). Their neighborhood structure, derived from the seminal work of Balas (1969), has laid the foundations for the most effective search strategies currently employed. The N1 neighborhood is generated by swapping any adjacent pair of critical operations on the same machine, and based on the following properties:

- Given a feasible solution, the exchange of two adjacent critical operations cannot yield an infeasible solution;
- The permutation of non-critical operations cannot improve the objective function and even may create an infeasible solution;
- Starting with any feasible solution, there exists some sequence of moves that will reach an optimal solution (known as the connectivity property)[1].

However, the size of the neighborhood N1 is quite large and includes a great number of unimproved moves. In order to reduce the number of block moves, Matsuo *et al*. (1988) proved that **unless the job-predecessor of u or the job-successor of v is on the critical path $P(0,n)$, the interchange containing u and v cannot reduce the makespan,** i.e. swapping internal operations within a block never gives an immediate improvement on the makespan. The work of Matsuo et al. (1988) allowed the neighborhood of moves to be reduced quite substantially.

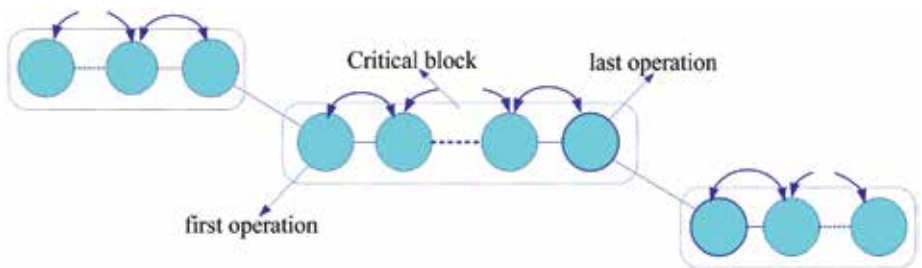


Fig. 3. The N1 neighborhood of moves

A neighborhood due to Grabowski *et al*. (1988), based on extending a neighborhood for a one machine problem (Grabowski *et al*. 1986), provides the next advance. This work introduced the concept of a block and defined a move to consist of inserting an operation at either the front or the rear of the critical block. Then, further refinements have been provided by Dell'Amico & Turbian(1993) (N4), Nowicki & Smutnicki (1996) (N5) and Balas & Vazacopoulos (1998) (N6).

The neighborhood N4 moves all operations i in a block to the very beginning or to the very end of this block, (Dell'Amico and Turbian proposed two neighborhood structures N3 and N4; in this paper we only discuss the neighborhood N4), N4 neighborhood structure is connected. The neighborhood N5 involves the reversal of a single border arc of a critical

1 However Kolonko (1998) proves that the connectivity property does not imply convergence to an optimum in these neighbourhoods.

block[2] and is substantially smaller (or more constrained) than the other neighborhoods, whereas the neighborhood N4 and N6 involves the reversal of more than one disjunctive arc at a time and thus could investigate a considerably larger neighborhood. The neighborhood N6, which is also considered as an extension of the neighborhood N5, is more constrained (smaller) than the neighborhood N4 and is currently one of the most efficient neighborhood structures. These neighborhoods N4, N5 and N6 are illustrated in Fig. 4, Fig. 5 and Fig. 6, respectively.

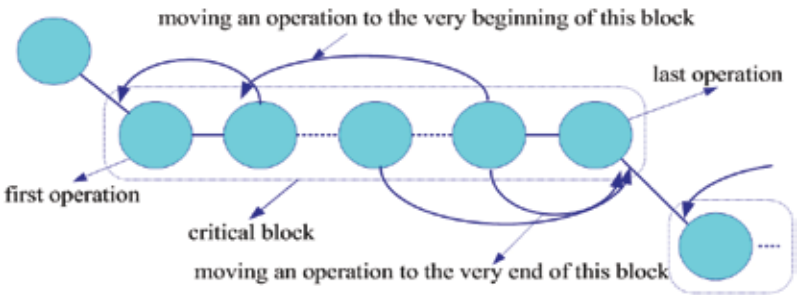


Fig. 4. The N4 neighborhood of moves

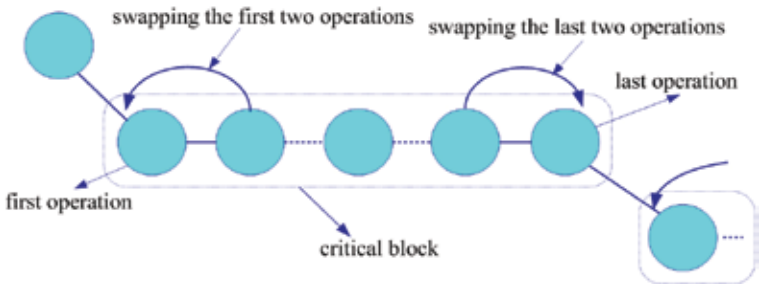


Fig. 5. The N5 neighborhood of moves

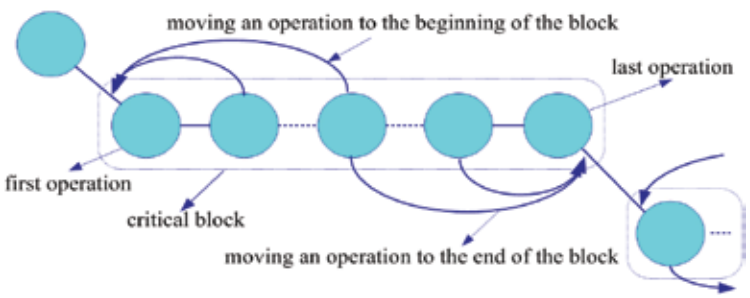


Fig. 6. The N6 neighborhood of moves

[2] In this neighborhood only one critical path is generated. A move is defined by the interchange of two successive operations *i* and *j*, where *i* or *j* is the first or last operation in a block that belongs to a critical path. In the first block only the last two operations and symmetrically in the last block of the critical path only the first two operations are swapped.

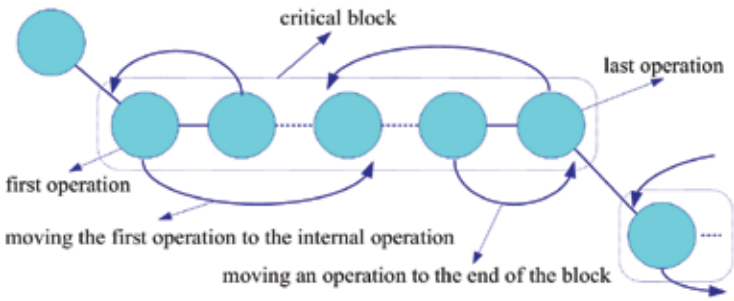


Fig. 7. The further extended neighborhood structure

According to Matsuo et al.(1988), we have seen that in order to achieve an improvement by an interchange on u and v (assume u is processed before v), either $JP[u]$ or $JS[v]$ must be contained on the critical path P (0, n), that is, either u or v must be the first or last operation of a critical block. Therefore, a further extended neighborhood used in this paper is proposed, **which not only inserts an operation to the beginning or the end of the critical block, but also moves the first or the last operation into the internal operation within the block,** illustrated in Fig. 7. This leads to a considerably larger neighborhood and investigates a much larger space. However, the questions to be explored are under what conditions an interchange on critical operation u and v is guaranteed not to create a cycle and how to reduce the neighborhood size. Next we give the two theorems and the proof for our neighborhood structure. Consider a feasible solution s:

Theorem 1.

If two operations u and v to be performed on the same machine are both on the critical path P (0, n) , and $L(v, n) \geq L(JS[u], n)$, then moving u right after v yields an acyclic complete selection.

Proof. By contradiction: suppose moving u right after v create a cycle C. Then C contains either $(u, JS[u])$ or $(u, MS[v])$. If $(u, JS[u]) \in C$, there is a path from $JS[u]$ to v in D_s(the cycle is $JS[u] \rightarrow v \rightarrow u \rightarrow JS[u]$), and hence $L(JS[u], n) > L(v, n)$, contrary to assumption. If $(u, MS[v]) \in C$, there is a path from $MS[v]$ to v in D_s, contrary to the assumption that D_s is acyclic. This completes the proof.

Theorem1 derives the idea that moving a critical operation u right after a critical operation v will not create a cycle if there is no directed path from $JS[u]$ to v in D_s. The Theorem1 could also be described briefly as follows: Given a feasible solution, if exchange two critical operation u and v, and the start time of the operation JS[u] is more than or equal to the start time of the operation v, then moving u right after v yields a feasible solution. We notice that if u and v are adjacent in critical path P (0, n), then the conditions described are always satisfied (This is N1 neighborhood structure).

By analogy, we have Theorem 2.

Theorem 2.

If two operations u and v to be performed on the same machine are both on the critical path P (0, n), and $L(0, u)+p_u \geq L(0, JP[v])+p_{JP[v]}$), then moving v right before u yields an acyclic complete selection.

Proof. Parallels that of theorem 1.

In order to construct our neighborhood structure, we in fact extend the scope of the Proposition2.2 and Proposition2.3 proposed by Balas & Vazacopoulos (1998) and present Theorem 1 and Theorem 2. Theorem 1 and Theorem 2 could be applied to an interchange on

any two *critical* operations u and v to be performed on the same machine, whether or not it contains either the $JS[u]$ or $JP[v]$ on the critical path. In our experiment, we observe that the neighborhood constructed by Theorem 1 and Theorem 2 is simpler and more constrained (smaller) than the similar neighborhood N4. Therefore, our search neighborhood could now be concisely defined as follows:
(1) If a critical path P (0, n) containing u and v also contains $JS[v]$, then insert u right after v and move v right before internal operations;
(2) if a critical path P (0, n) containing u and v also contains $JP[u]$, then insert v right before u and move u right after internal operations.
Assume that an interchange on two operation u and v results in a makespan increase of the new schedule compared to the old one. Then it is obvious that every critical path in new schedule contains the arc (v, u) (Balas & Vazacopoulos, 1998). We could make use of the fact and further reduce our neighborhood size.

4. The framework of the hybrid of TS and SA

Tabu search, defined and developed primarily by Glover (1989, 1990), has been successfully applied to a large number of combinatorial optimization problems, especially in production scheduling domain. Tabu search is an enhancement of the hill climbing heuristic. In order to avoid cycling through previous solutions, a short-term memory structure known as the tabu list is implemented. According to Brucker (Brucker, 1995), tabu search is an intelligent search that uses a memory function in order to avoid being trapped at a local minimum. Its goal is to emulate intelligent uses of memory, that is, tabu search tries to create memory itself similar to use of some memory functions of people in order to find its way out.
Tabu search algorithm was first applied to the JSSP by Taillard. Since then, researchers have introduced numerous improvements to Taillard's original algorithm. Among these tabu search methods, algorithm TSAB developed by Nowicki & Smutnicki (1996) introduces the significant breakthrough in both effectiveness and efficiency for the JSSP. However, even for the famous TSAB algorithm, the choice of an initialization procedure has an important influence on the best solution found, and a better initial solution might provide better results (Jain et al., 2000). By contrast, simulated annealing is not a powerful technique for the JSSP, but the initial solution has little influence on the solution quality obtained by simulated annealing procedure. However, due to lack of the memory function, simulated annealing may return to old solutions and become oscillation in local optimum surrounding. This causes the search to consume excessive computational times. Therefore, the combination of the memory function (avoid cycling) of tabu search and the convergent characteristics of simulated annealing provide the rationale for developing a hybrid TS/SA strategy to solve the combinatorial optimization problems. Therefore, in this paper, by exploiting the properties of the JSSP and the complementary strengths and weaknesses of the two paradigms, we present the newly designed hybrid TSSA algorithm.
The idea of the hybrid approach for the JSSP is based on the two important observations, due to Nowicki & Smutnicki (2001), as follows. First, the space structure of the considered job shop problem owns big valley (BV) and the best elite solutions dispersed over BV area. Second, tabu search is perfectly attracted to big valley area. Even though the initial solution was set relatively far from the valley, elite solutions generated by tabu search can still be collected inside big valley. The two observations indicate that tabu search is suitable for finding good solutions inside the big valley, and these good solutions previously

encountered provide better starting points for further space exploration than various initial solutions do. However, the number of solutions inside big valley is so large that it is unrealistic to expect that the whole valley might have been exhaustively searched. Nevertheless, simulated annealing, which possesses good convergence properties and accepts the candidate solution probabilistically by the *Metropolis acceptance criterion*, provides a procedure to find the sufficient "good" solutions over the big valley. Therefore, our hybrid TSSA algorithm, which simulated annealing is used to find the promising elite solutions inside big valley generated on the search history and tabu search intensifies search around the solutions, is proposed. The main idea of TSSA algorithm is also related to the strategy that designs the more efficient forms of finite convergent tabu search based on recency-memory (Glover et al., 2002).The general framework of hybrid TSSA algorithm for the JSSP is outlined in Fig. 8.

Step 1. Generate an initial solution s and calculate its makespan $f(s)$, set the current solution $s^* = s$, the best solution $s^b = s$, the best makespan $f(s^b) = f(s)$, $iter = 0$ and the initial temperature $T = t_0$, empty tabu list and push the s^* onto the elite solution stack L (LIFO list).
Step 2. Set $iter = iter+1$, generate neighbors of the current solution s^* by a neighborhood structure. If the s^* is optimal, then stop.
Step 3. Select the best neighbor which is not tabu or satisfies the aspiration criterion, and store it as the new current solution s^*. Update the tabu list.
Set the temperature T, and calculate the exact makespan $f(s^*)$.
If ($f(s^*) < f(s^b)$ || $\exp(f(s^b) - f(s^*))/T >$ random [0, 1])
{
"Push" the s^* onto the elite solution stack L.
}
Step 4. If $f(s^*) < f(s^b)$, then set $s^b = s^*$, $f(s^b) = f(s^*)$, $iter = 0$.
Step 5. If $iter \le ImproveIter$ then go to Step 2.
Step 6. If a termination criterion is satisfied then stop. Else "pop" a solution from the elite solution stack L, shift the solution to active schedule and store the active solution as the current solution s^*, set $iter = 0$ and empty tabu list, then go to Step 2.

Fig. 8. Outline of TSSA algorithm for the JSSP

In the hybrid TSSA algorithm, the core tabu search is a straightforward implementation of tabu search intensification strategy. A strong diversification strategy using simulated annealing procedure to find the elite solutions inside big valley is equipped with the core tabu search and directs the intensified search to other regions of the solution space. More precisely, starting from a randomly initial solution, TSSA algorithm executes the core tabu search procedure and tracks the sufficiently "good" solutions found by simulated annealing procedure on the search history. The "good" solutions found by simulated annealing procedure are stored in the elite solution stack L. Each new good solution is "pushed" onto the solution stack L when it is discovered. Subsequently, such solutions may be "popped" from the stack L in turn as new incumbent solutions, from which an intensified search is performed in a pre-specified number of iterations (*ImproveIter*). Given the suitable temperature T, the solutions in the elite solution stack should not be exhausted. The algorithm terminates when the total number of iterations reaches to the given value or the solution is proved to be optimal.

It can be seen that the hybrid TSSA framework in Fig. 8 can be converted to the traditional tabu search by omitting the simulated annealing unit, whereas it can be converted to a general simulated annealing by setting the length of tabu list to zero and the length of solution stack L to one. Such hybrid TS/SA strategy retains advantages of tabu search and simulated annealing, and provides a promising methodology to solve the other computationally intractable problems. For different problems, the neighborhood structure, parameters and algorithm criteria should be designed appropriately. Due to utilizing the properties of the JSSP, the new hybrid algorithm is closer resemblance to tabu search. According to the characteristics of the different problems, it even may develop the different TS/SA framework, such as the SATS strategy which most closely resembles simulated annealing and incorporates tabu search into simulated annealing. In the next section, a detailed description of each component function of TSSA algorithm for the JSSP is provided.

5. The implementation of TSSA

5.1 Initial solution

The initial solution can be generated by various methods such as the priority dispatching rules, the insertion algorithm, the shifting bottleneck procedure and random methods. Empirical testing shows that the initial solution methods affect the solution quality for tabu search algorithm, so that the better initial solution might provide better results. Thus, for the majority of the tabu search, the specialized initialization procedure is used to obtain the better initial solution. However, the initial solutions have little influence on the solution quality provided by TSSA algorithm, but are effective on the running time. Hence the search is initiated from the randomized active solution.

5.2 The neighborhood structure

Neighborhood structure and move evaluation strategies are directly effective on the efficiency of the search for the JSSP, and unnecessary and infeasible moves must be eliminated if it is possible. Currently, the most well-known neighborhood structures are all based on the concept of blocks. In the TSSA algorithm, taking account on a balance of the effectiveness and efficiency, if the number of operations is less than 200, then N6 neighborhood structure introduced by Balas and Vazacopoulos (1998) is adopted; otherwise, the neighborhood structure proposed in this paper is applied.

5.3 Move evaluation

The run-time of local search algorithm for the JSSP is typically dominated by the cost of computing each move. Therefore, in order to make the algorithm more efficiently, a number of neighborhood evaluation strategies, such as exact methods and estimation methods, have been proposed. The procedure suggested by Ten Eikelder et al.(1999), called *bowtie*, is the most efficient exact method for the JSSP, which only recalculates the head and the tail values of the operations that need to be updated after the move. However, this exact method still takes a very long computational time for the larger instances. In order to accelerate the search process, the estimation methods which can quickly filter out moves that have a high probability of directing the search to new elite solution have been proposed. A fast estimation approach has been presented by Taillard (1994). A further accurate approach of this strategy has also been proposed by Nowicki and Smutnicki (2002), which is employed

in the famous *i*-TSAB algorithm. But these estimation strategies are only adapted to swap the adjacent operations of the critical block. A significant estimation strategy proposed by Balas and Vazacopoulos (1998) could be applied to reverse more than one disjunctive at a time, and the procedure of Balas and Vazacopoulos is also one of the most efficient implementations to solve the JSSP. The procedure of the estimation strategy proposed by Balas and Vazacopoulos is given as follows.

$L(i, j)$ and $L^{u,v}(i, j)$ denoted as the length of a longest path from i to j (if it exists) before and after an interchange on u and v, respectively, and by $\lambda^{u,v}(i, j)$ our evaluation (estimate) of $L^{u,v}(i, j)$, namely

$$\lambda^{u,v}(0, n) = \max\{\lambda^{u,v}(0, w)+\lambda^{u,v}(w, n):\ w\in Q\} \tag{1}$$

where $Q := \{u, l_1, \ldots, l_k, v\}$ is the segment of the critical path $P(0,n)$ containing u and v. Here the estimates $\lambda^{u,v}(0, w)$ and $\lambda^{u,v}(w, n)$ are calculated recursively as follows.

Case 1. The interchange on u and v is a forward one, then we have

$$\lambda^{u,v}(0,l_1) = \begin{cases} L(0,JP[l_1]) + p_{JP[l_1]} & \text{if } u \text{ was the first operation on its machine,} \\ \max\{L(0,JP[l_1]) + p_{JP[l_1]}, L(0,MP[u]) + p_{MP[u]}\} & \text{otherwise;} \end{cases}$$

for $w\in\{l_2, \ldots, l_k, v\}$,

$$\lambda^{u,v}(0,w) = \max\{L(0,JP[w] + p_{JP[w]}, \lambda^{u,v}(0,MP[w]) + p_{MP[w]}\},$$

and

$$\lambda^{u,v}(0,u) = \max\{L(0,JP[u] + p_{JP[u]}, \lambda^{u,v}(0,v) + p_v\}$$

Further, we have

$$\lambda^{u,v}(u,n) = \begin{cases} p_u + L(JS[u],n) & \text{if } v \text{ was the last operation in its machine,} \\ p_u + \max\{L(JS[u],n), L(MS[v],n)\} & \text{otherwise,} \end{cases}$$

$$\lambda^{u,v}(v,n) = p_v + \max\{L(JS[v],n), \lambda^{u,v}(u,n)\},$$

and for $w\in\{l_1, \ldots, l_k\}$,

$$\lambda^{u,v}(w,n) = p_w + \max\{L(JS(w),n), \lambda^{u,v}(MS(w),n)\}.$$

Case 2. The interchange on u, v is a backward one. Then we have

$$\lambda^{u,v}(0,v) = \begin{cases} L(0,JP[v]) + p_{JP[v]} & \text{if } u \text{ as the first operation in its machine,} \\ \max\{L(0,JP[v]) + p_{JP[v]}, L(0,MP[u]) + p_{MP[u]}\} & \text{otherwise,} \end{cases}$$

$$\lambda^{u,v}(0,u) = \max\{L(0,JP[u]) + p_{JP[u]}, \lambda^{u,v}(0,v) + p_v\},$$

and for $w\in\{l_1, \ldots, l_k\}$,

$$\lambda^{u,v}(0,w) = \max\{L(0,JP[w]) + p_{JP[w]}, \lambda^{u,v}(0,MP[w]) + p_{MP[w]}\}$$

Further, we have

$$\lambda^{u,v}(l_k,n)=\begin{cases} p_{l_k}+L(JS[l_k],n) & \text{if } u \text{ was the first operation on its machine,} \\ p_{l_k}+\max\{L(JS[l_k],n),L(MS[v],n)\} & \text{otherwise;} \end{cases}$$

for $w \in \{u, l_2, \ldots, l_{k-1}\}$,

$$\lambda^{u,v}(w,n)=p_w+\max\{L(JS[w],n),\lambda^{u,v}(MS[w],n)\},$$

And

$$\lambda^{u,v}(v,n)=p_v+\max\{L(JS[v],n),\lambda^{u,v}(u,n)\}$$

In this paper, we compare the exact approach of Ten Eikelder et al. with the estimation approach of Balas and Vazacopoulos using our algorithm. The empirical testing shows that the estimation approach reduces the computational effort by 10 to 20 % in comparison with the exact approach, and the efficiency of the search increases as the instances become larger. Intuition suggests that exact approach might perform better than the estimation approach in the solution quality, but empirical results indicate that this intuition remains unconfirmed.

5.4 Tabu list and tabu status of move

The basic role of tabu list is to avoid the search process turning back to the solutions visited in the previous steps. The elements stored in the tabu list are the attributes of moves, rather than the attributes of solutions. The main purpose of using this attributive is to reduce the computational cost. A side effect of implementing the "a partial attribute" tabu list is that it may lead to giving a tabu status to unvisited solution or even an interesting solution. However, an aspiration criterion, which accepts the move provided that its makespan is lower than that of the current best solution found so far, is used by tabu search algorithm to avoid this problem. With our neighborhood structure, the move selected at each step may reverse more than one disjunctive arc and involve a sequence of operations. Unlike the majority of the previous tabu searches for the JSSP which only store the sequence of the operations exchanged in the tabu list, tabu search algorithm in this paper stores not only the sequence of the operations, but also their positions on the machine. This approach could better represent the attributes of moves. More precisely, if a move consists of the exchange on operations u and v, then a move achieved the same sequence of operations and positions (namely, the operations from u to v (u,..,w,..,v) and their positions on the machine from u to v (p_u,..,p_w,..,p_v)) is not permitted for the duration that the move is recorded as "tabu".

The length of the tabu list determines the time limit of the moves remaining on the memory, which is discouraged at the current iterations. Therefore, the length of the tabu list plays an important role in the search process. Moreover, if the length of list is too short cycling cannot be avoided; conversely, a too long size creates too many restrictions and influences the intensification of the search. It has been observed that the average number of the visited solutions grows with the increase of the tabu list size. How to effciently set the length of tabu list in JSSP is still open problems, and setting the parameters often suffers from tedious trial and error. An empirical study suggests it may be possible to obtain superior results when the length of tabu list is allowed to vary dynamically during the course of the search. For example, Taillard (1994) suggests the length of the tabu list be randomly selected from a range between given minimal and maximal values and changed each time a number of iterations. Another example is the somewhat more sophisticated approach of Dell'Amico and Trubian (1993).

Therefore, dynamic tabu list is applied and the length of tabu list is randomly chosen between two given minimal and maximal values [L_{min}, L_{max}]. Our preliminary experiments show that the suitable length of tabu list increases as the ratio of the number of jobs (n) to the number of machines (m) becomes larger for the considered problem. The smallest length of tabu list could be set to $L = 10+n/m$ for getting good results, and $L_{min} = [L]$, $L_{max} = [L+2]$ are appropriate values for the JSSP.

5.5 The recovery of the elite solutions based on the simulated annealing

In this paper, a powerful recency-based memory mechanism, which utilizes simulated annealing to find the elite solutions inside big valley, is built in the core tabu search and induces the search to pursue different trajectories. The recency-based memory mechanism is adjusted as follows. If the current solution satisfies:

$$f(s^*) < f(s^b) \text{ or } \exp(f(s^b) - f(s^*))/T > \text{random } [0, 1] \quad (2)$$

f (s^*) and f (s^b) stand for the makespans of the current solution and the best solution, respectively, then the solution is pushed onto the elite solution stack L. However, if a pre-specified number of iterations (*ImproveIter*) have been executed without an improvement in the best-so-far solution, the solution on top of the solution stack L is popped, shifted to the active schedule and installed as the current solution. The TSSA algorithm then reinitiates the core tabu search procedure from the new current solution, as well as resets the original parameters and clears the tabu list.

During the run of the TSSA, the elite solutions found by simulated annealing are stored in the elite solution stack L. The maximum number of the solution stack L is a fixed number and denoted as *Maxelite*. For the test instances, a *Maxelite* of 30 is found to offer an appropriate value. Temperature T of simulated annealing has an important influence on the selected elite solutions and consequently affects the quality of solution provided by TSSA algorithm. If the temperature is too low, the algorithm may be terminated earlier due to the elite solution stack being quickly exhausted, whereas if the temperature is too high, it can not guarantee that the elite solutions are effectively selected. Empirical testing shows that the suitable temperature T increases as the instances size becomes larger. For the general JSSP instances, T can be set to value 2-6 according to the instance size. It can be set to $T = bestMakespan/Temp$, where *bestMakespan* is the best makespan found so far and *Temp* means a fixed parameter based on the instance size. It can be seen that the temperature decreases as the best makespans being found, so simulated annealing performs a "fine" search around local optima. In addition, it must be noted that the solutions that run during the course of the tabu search only guarantee the semi-active schedules, not the active schedules. However, the optimal schedule is in the set of the active schedules. Therefore, the TSSA algorithm converts the semi-active schedules popped from the elite solution stack into active schedules. This approach could better direct the search to explore new promising regions and hopefully increase the chances of finding the global optimum.

5.6 Move selection

The choice rule of the tabu search method is to select the move which is non tabu with the lowest makespan or satisfies the aspiration criterion. Nevertheless, a situation may arise where all possible moves are tabu and none of them satisfy the aspiration criterion. In such a case, one might use the oldest tabu move, or randomly select a tabu move. Empirically, the second strategy that randomly selects a move among the possible moves proves better and is implemented in the TASA algorithm.

5.7 Cycle check

TSSA algorithm employs a simple and fast mechanism similar to TSAB, with the exception of setting the *cycle_gap* to fixed numbers, to detect the cycling behavior. The detailed contents can be seen by Nowicki & Smutnicki (1996) and Jain et al. (2000). When a cycle is detected, instead of continuously intensifying search to the current solution by the N6 neighborhood, we apply the N1 neighborhood structure to yielding a small perturbation to the current solution. This mechanism is able to remain nearby the current solution while simultaneously inducing search to escape from the cycling.

5.8 Termination criterion

The algorithm stops when it has performed a given total number of iterations (*TotIter*), or the elite solution stack has been exhausted, or the solution is proved to be optimal. If one of the following conditions is satisfied: (1) All critical operations are processed on the same machine (i.e. only one critical block is generated) or belong to the same job (i.e. each block consists of only one operation); (2) The makespan is equal to the known lower bound, then the solution is optimal and the algorithm is terminated. In addition, the elite solution stack should not be exhausted provided that the temperature T is suitably set.

6. Computational results

The proposed algorithm above was implemented in VC++ language on a personal computer Pentium IV 3.0G. In order to evaluate and compare the performance of this algorithm, we tested it on the well-known benchmark problems taken from literature. These job shop scheduling instances include the following classes:

(a) Three instances denoted as FT6, FT10, FT20 (size $n\times m$ = 6×6, 10×10, 20×5) due to Fisher and Thompson (1963), forty instances LA01-40 (size $n\times m$ = 10×5, 15×5, 20×5, 10×10,15×10, 20×10, 30×10, 15×15) due to Lawrence (1984), five instances ABZ5-9 (size $n\times m$ = 10×10, 20×15) due to Adams et al. (1988) and ten instances ORB01-10 (size $n\times m$ = 10×10) due to Applegate & Cook (1991).

(b) Four instances denoted as YN1-4 (size $n\times m$ = 20×20) due to Yamada & Nakano (1992) and twenty instances SWV01-20 (size $n\times m$ = 20×10, 20×15, 50×10, 50×10) due to Storer et al. (1992).

(c) Eighty instances denoted as DMU01-DMU80 (size $n\times m$= 20×15, 20×20, 30×15, 30×20, 40×15, 40×20, 50×15, 50×20) due to Demirkol et al(1998).

The FT, LA, ABZ, SWV and YN problems are available from the OR Library site http://www.ms.ic.ac.uk/job/pub/jobshop1.txt, while the DMU problems are available from http://gilbreth.ecn.purdue.edu/~uzsoy2/benchmark/problems.html. For these benchmark set, the best known upper bounds (UB_{best}) and the best known lower bounds (LB_{best}) are taken from Jain et al. (1999) and updated with the improved results from Nowicki & Smutnicki (2002).

To analyze the quality of the solutions, the mean relative error (MRE) was calculated from the best known lower bound (LB_{best}), and the upper bound (UB_{solve}) that is the makespan of the best solution solved by our algorithm, using the "relative deviation" formula $RE = 100\times(UB_{solve} - LB_{best}) / LB_{best}$ for each instance. Due to the stochastic properties of this algorithm, it is not reasonable to compare the best makespan MRE (b-MRE) of TSSA with the results of the other algorithms. Therefore, in order to fairly evaluate the performance of TSSA algorithm, we compared the mean performance (av-MRE) of TSSA with the results of the other algorithms.

In the remainder of this section, the first section gives the comparison of the neighborhood strategies and the move evaluation strategies respectively, in order to analyze the new neighborhood structure and the estimation strategy used in this paper. In the second section the proposed TSSA algorithm is performed on a large number of the benchmark instances to measure the performance of this algorithm.

6.1 Comparison of the neighborhood structures and move evaluation strategies

In order to compare the neighborhood structures and move evaluation strategies effectively, we set the length of elite solution stack L (*Maxelite*) to 0, and convert the TSSA algorithm to the traditional tabu search algorithm.

(1) Comparison of the neighborhood structures

Six neighborhood structures are tested and compared below. They are denoted by NS1 (van Laarhoven et al. 1992, N1), NS2 (Chambers & Barnes, 1994), NS3 (Nowicki & Smutnicki 1996, N5), NS4 (modify the Dell'Amico & Trubian, 1993, N4), NS5 (Balas & Vazacopoulos 1998, N6). Finally, NS6 denotes our new neighborhood structure. Meanwhile, NS1, NS2, NS3 and NS5 are well-known neighborhoods in literature, and these neighborhoods are all based on the concept of block except for N1. For further information, see Section3 and Blażewicz et al. (1996). NS4 slightly differs from N4 introduced by Dell'Amico and Trubian. NS4 neighborhood moves the operations in a block to the beginning or the end of this block according to the procedure of Dell'Amico & Trubian (1993), and the moves do not allow for an interchange on u and v when the critical path containing u and v contains neither $JS[u]$ nor $JP[v]$.

In order to evaluate these neighborhood structures exactly, we use tabu search algorithm as the platform, and the difference only exists among the neighborhood structures. The initial solution is generated by SPT priority dispatch rule, and the length of tabu list is set to 12 suggested by Geyik & Cedimoglu (2004) except for NS1 which applies the approach of Taillard. Each move in the neighborhood is evaluated exactly. The algorithm is terminated when the number of disimproving moves reaches to the value 3000. The benchmark set FT, LA, ABZ, SWV and YN containing 72 instances are tested. Moreover, several measures that gain some statistics relating to the comparison are presented. They are the mean makespan C_{max} (MC_{max}), the number of the solution found equal to the known best solution (NBE), the mean number of evaluated neighbors (MEN), the mean number of iterations (MNI) and the total CPU time performed in all instances (CPU-time).

	NS6	NS5	NS4	NS3	NS2	NS1
MC_{max}	1416	1420	1415	1462	1473	1600
MRE	3.31	3.51	3.45	5.52	5.84	12.6
NBE	32	33	32	26	24	20
MEN	106163	70216	127140	48973	48510	107649
MNI	5731	5468	5619	6690	6206	4149
CPU-time	309	205	462	121	134	307

Table 2. The comparison of the six neighborhood structures

Table 2 summarizes the computational results relating to the six neighborhoods. NS6 offers the minimum MRE value among the six neighborhood structures. The MRE provided by NS4 (better than the original N4) is close to that of NS6, but NS4 consumes too much

computing times. NS5 is more effective than NS3, NS2 and NS1, and NS3 is better than NS2 and NS1. Overall, it can be seen that NS6 is an effective neighborhood structure for the JSSP.

(2) Comparison of the move evaluation strategies

The estimation approach proposed by Balas and Vazacopoulos and the exact evaluation approach suggested by Ten Eikelder et al. are compared using the tabu search algorithm as the test platform, see Table 3. The two test platforms are all similar except for the move evaluation methods. To make a full statistical comparison, all sets of benchmarks applied in this paper are selected and used to test. Table 3 presents the results of the estimation approach (Balas and Vazacopoulos) and the exact approach (Ten Eikelder et al.) when initiated from SPT priority rule.

	MC_{max}	MRE	NBE	MEN	MNI	CPU-time	Tabu list size
estimation approach	3340	4.42	40	215598	10445	367.2	12
exact approach	3339	4.44	41	234598	10785	2206.8	12

Table 3. Comparison of the estimation approach (Balas) with the exact approach (Ten Eikelder)

The empirical results in Table 3 indicate that the estimation approach of Balas and Vazacopoulos is about 5-6 times faster on average in evaluating moves in comparison to the exact evaluation of Ten Eikelder et al. In the experiments, the estimation approach performs better than the exact approach, by not only getting to the solutions faster, but also having no effect on the solution quality. For example, these two evaluation strategies provide the approximately similar value of MRE. However, it can be seen from Table 3 that the estimation approach need the relatively "small" MNI (the mean number of iterations) to achieve the similar value of MRE. A plausible explanation for this is that applying the estimation strategy leads to the algorithm to intensify search in the visited region and mitigates the drawback of the estimation approach. Therefore, the estimation approach proposed by Balas and Vazacopoulos is implemented to perform these computations of each move in the following section.

6.2 TSSA algorithm for the JSSP

We compared the TSSA algorithm with the best approximation algorithms which provide the detailed computational results, and used the following notation for those algorithms: TSAB stands for the Taboo Search of Nowicki & Smutnicki (1996), BV stands for the Guided local Search with Shifting Bottleneck of Balas & Vazacopoulos (1998). Meanwhile, SB-RGLS5 stands for the solution of SB-RGLS5 procedure of Balas & Vazacopoulos (1998) and BV-best stands for the best solution obtained by Balas & Vazacopoulos. TSSB stands for a tabu search method guided by shifting bottleneck of Pezzella & Merelli (2000). Among these papers listed above, the algorithms TSAB, BV and TSSB provide the detailed makespan and running time of each instance.

TSSA algorithm offers a very short running time within several minutes (even seconds) on our personal PC for the general hard instances, and it requires about ten minutes for the particularly hard instances. Nevertheless, empirical studies of processor speeds show that it

is hard to get the real computer-independent CPU time. Hence, in this paper we enclosed for each algorithm the original name of machine and the original running time to avoid discussion about the different computers speed used in tests.

TSSA algorithm was initiated from the active solution randomly generated. Parameters of the length of tabu list, *ImproveIter* (limit on umimproved iterations), *T* (Tempeature) and *TotIter* (the total number of the iterations) were chosen experimentally in order to ensure a compromise between efficiency and effectiveness. The choice of the length of tabu list can be seen in the Section 5.4. Parameter *ImproveIter* was bounded by the given minimal and maximal values between 2500-5000 depending on the number of jobs (n) and the number of machines (m). More precisely, if the result of $10\times n\times m$ is between 2500-5000, then *ImproveIter* = $10\times n\times m$. Otherwise, if the result of $10\times n\times m$ is less than 2500 or more than 5000, then *ImproveIter* = 2500 or *ImproveIter* = 5000 respectively. For the test instances, we set *T* = *bestMakespan*/*Temp*, see Section 5.5. Meanwhile, if the number of operations is less than or equal to 400, *Temp* = 300. Otherwise, *Temp* = 300+50×n/m. The parameter *TotIter* has an influence on the running time and solution quality of this scheme. The increasing of *TotIter* yields a higher possibility of obtaining a high-quality solution. However, a further increase of this parameter has little influence on the makespan but evidently increases the running time. In order to ensure the balance between the running time and solution quality, our preliminary experiments show that *TotIter* can be set to $n\times 100000/2$ for general hard instances, but for the particularly hard instances it increases correspondingly for the sake of finding a better solution.

During various tests (with tuning parameters) and standard tests, TSSA algorithm found some new upper bounds. The best upper bounds may be found after the running of many times, but standard tests only ran ten times to get the average makespan and running time of each instance.

(1) Results for instances (a)

In this section we discuss the behavior of TSSA on the four oldest benchmarks: FT6,10,20, ABZ5-9, LA01-40 and ORB01-10. The number of their operations ranges from 55 to 300. Despite their relatively small size, these instances were very hard to solve. For example, FT10 remained unsolved until twenty years later. However, by years, these instances have been solved optimally except for ABZ8 and ABZ9, some of them by the B&B scheme, some by approximate algorithms.

Firstly, the common benchmarks FT, LA and ABZ are tested by TSSA algorithm. The problems FT20(20×5), LA01-05(10×5), LA6-10(15×5), LA11-15(20×5) and LA30-35 (30×10) are relatively easy because the number of jobs is several times larger than the number of machines. They could be solved to optimality by TSSA algorithm in a second, so their results are omitted from the table. Table 4 shows the comparison of the performance of TSSA algorithm with those of TSAB, BV-best and TSSB. In this table, it lists the best MRE (b-MRE), the average MRE (av-MRE) and the average running time of each group (T_{av}) of each algorithm. We enclose the original running time and original machines reported by the authors of TSAB, BV-best and TSSB in Table 4 (similar to Table 5-8 below). It can be seen that TSSA algorithm performs very quickly and acquires the results in only half a minute on average on the personal PC. On these problems, the av-MRE of TSSA is clearly lower than the MRE of TSAB and TSSB, and the b-MRE of TSSA is also better than the b-MRE of BV-best.

Problem group	Size	TSSA b-MRE	TSSA av-MRE	TSSA T_{av}(s)	TSAB[a] MRE	TSAB[a] T_{av}(s)	BV-best[b] b-MRE	BV-best[b] T_{av}(s)	TSSB[c] MRE	TSSB[c] T_{av}(s)
LA01-05	10×5	0.00	0.00	0.0	0.00	3.8	0.00	3.9	0.00	9.8
LA16-20	10×10	0.00	0.00	0.2	0.02	68.8	0.00	25.1	0.00	61.5
LA21-25	15×10	0.00	0.03	13.6	0.10	74	0.00	314.6	0.10	115
LA26-30	20×10	0.02	0.02	15.2	0.16	136.4	0.09	100.0	0.46	105
LA36-40	15×15	0.03	0.19	36.1	0.28	375.6	0.03	623.5	0.58	141
ABZ5-6	10×10	0.00	0.00	2.7	0.08	16.5	0.00	252.5	0.00	77.5
ABZ7-9	20×15	2.10	2.80	88.9	4.34	−	2.45	6680.3	3.83	200
Average		0.31	0.43	22.3	0.71	−	0.37	1142.8	0.71	101.4

[a] the CPU time on the personal computer AT386DX.
[b] the CPU time on the SUN Sparc-330.
[c] the CPU time on personal computer Pentium 133MHz.

Table 4. Comparison with the other three algorithms for LA and ABZ instances

Problem	Size	UB(LB)	TSSA Best	TSSA M_{av}	TSSA T_{av}(s)	TSAB	BV SB-RGLS5	BV BV-best	TSSB
FT10	10×10	930	930*	930	3.8	930	930	930	930
LA19	10×10	842	842*	842	0.5	842	842	842	842
LA21	15×10	1046	1046*	1046	15.2	1047	1046	1046	1046
LA24	15×10	935	935*	936.2	19.8	939	935	935	938
LA25	20×10	977	977*	977.1	13.8	977	977	977	979
LA27	20×10	1235	1235*	1235	11.7	1236	1235	1235	1235
LA29	20×10	1152	1153	1159.2	63.9	1160	1164	1157	1168
LA36	15×15	1268	1268*	1268	9.9	1268	1268	1268	1268
LA37	15×15	1397	1397*	1402.5	42.1	1407	1397	1397	1411
LA38	15×15	1196	1196*	1199.6	47.8	1196	1196	1196	1201
LA39	15×15	1233	1233*	1233.8	28.6	1233	1233	1233	1240
LA40	15×15	1222	1224	1224.5	52.1	1229	1224	1224	1233
ABZ7	20×15	656	658	661.8	85.9	670	664	662	666
ABZ8	20×15	665(645)	667	670.3	90.7	682	671	669	678
ABZ9	20×15	679(661)	**678**[d]	684.8	90.2	695	679	679	693
MRE			0.43	0.67	−	1.04	0.61	0.53	1.09

* The best solutions found by our algorithm are equal to the best known lower bounds
[d] The best makespans our algorithm found are better than the best previously known values

Table 5. Results for the fifteen tough instances

To make a more detailed performance comparison of the TSSA algorithm with the other algorithms, we select the fifteen most difficult instances among FT, LA and ABZ benchmarks. The majority of the 15 instances have been viewed as computational challenges, and even the optimal solutions of the ABZ8 and ABZ9 instances remain unknown until now. Table 5 shows the makespan performance statistics of each algorithm for the fifteen difficult problems, and Fig. 9 illustrates the gantt chart of the optimum solution for LA38 (15×15).

In this table, the column named UB (LB) lists the best known upper bounds(lower bounds) indicates in Jain et al. (1999), the next columns named Best, M_{av}, T_{av} show the best makespan, average makesan and average computing time in seconds obtained by TSSA algorithm over 10 runs respectively, and the last four columns show the results of the TSAB, SB-RGLS5, BV-best and TSSB. The last line shows the mean relative error (MRE) in order to analyze the effectiveness of these algorithms.

Table 5 shows that the av-MRE provided by TSSA is lower than the MREs of TAAB and TSSB, and is close to the MRE of SB-RGLS5. The b-MRE provided by TSSA is better than that of BV-best. Moreover, TSSA finds the optimal solution for the notorious instance FT10 almost every time within four seconds on average. Even for the general 15×15 instances, for example LA36 instance, TSSA has the capability of finding the optimal solution every time only in less than ten seconds on average. It must be pointed out that, unlike the majority of algorithms which are initiated from the better initial solution generated by the specialized methods, TSSA algorithm obtains these results from randomly initial solution. This indicates that TSSA algorithm is very robust and efficient. Among the fifteen instances, TSSA found the optimal solutions of ten (out of thirteen instances whose optimal solution values are known), and improved one upper bound among two unsolved instances, namely: ABZ9—678.

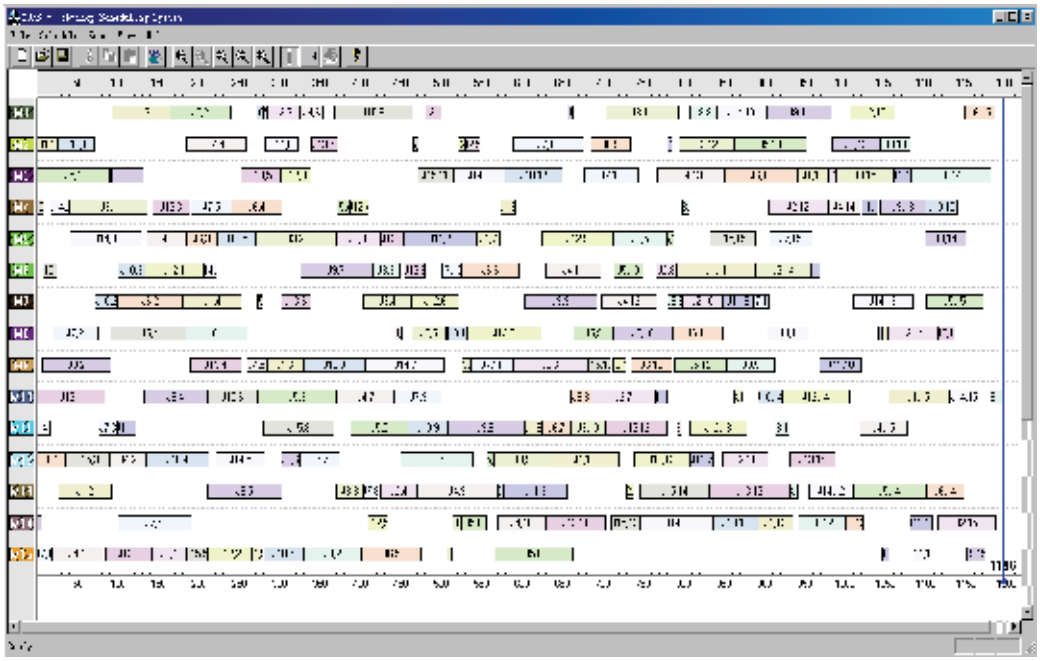


Fig. 9. Gantt chart showing the optimum solution for LA38 (15×15)

Finally, the ORB class which contains 10 instances is analyzed. Table 6 lists the detailed results of comparison. In this table, TSSA is clearly better than TSAB, BV-best and TSSB in terms of the solution quality. For example, TSSA found the optimal solutions for all instances, whereas BV-best, which is the best algorithm among the other three algorithms, found the optimal solutions for eight out of ten problems.

Problem	Size	LB	TSSA			TSAB '		BV-best[b]		TSSB[c]	
			Best	M_{av}	T_{av}(s)	Makespan	CI-CPU	Makespan	CPU(s)	Makespan	CPU(s)
ORB01	10×10	1059	1059*	1059	3.5	1059	548	1059	17.3	1064	82
ORB02	10×10	888	888*	888.1	6.4	890	376	888	88.4	890	75
ORB03	10×10	1005	1005*	1012.5	13.8	1005	356	1005	16.2	1013	87
ORB04	10×10	1005	1005*	1008.3	14.3	1011	427	1013	285.6	1013	75
ORB05	10×10	887	887*	888.6	6.6	889	389	889	15.2	887	81
ORB06	10×10	1010	1010*	1010	8.5	1013	472	1010	124.8	–	–
ORB07	10×10	397	397*	397	0.5	397	642	397	69.2	–	–
ORB08	10×10	899	899*	902.5	7.2	913	568	899	97.6	–	–
ORB09	10×10	934	934*	934	0.4	941	426	934	73.2	–	–
ORB10	10×10	944	944*	944	0.3	946	667	944	14.2	–	–
ORB01-10			0.0	0.17	6.2	0.37	487.1	0.10	80.2	0.46	80

Table 6. Results for ORB01-10 instances

(2) Results for instances (b)

YN class contains 4 instances with size 20×20, and no optimal solutions have been known. SWV class contains 20 instances with the number of operations between 200 and 500, and nine of them have not been solved for the optimal solutions. Benchmarks YN and SWV have not been tested by algorithm TSSB. Therefore we primarily compare TSSA with SB-RGLS5 and BV-best.

Table 7 shows the detailed results of comparison for YN instances. The solution quality TSSA algorithm provides is clearly better than that of BV in a short time. For example, TSSA algorithm achieves av-MRE = 6.99% within two minutes on our personal PC, whereas BV-best needs approximately 150 minutes on the SUN SParc-330 to achieve the similar aim. Moreover, TSSA algorithm found two new upper bounds, namely: YN1—884 and YN2—907. Fig. 10 illustrates the gantt chart of the best solution whose makespan is equal to 884 for YN1.

Similarly as for YN instances, the detailed results for SWV instances are shown in Table 8. In order to find a better solution, *TotIter* is set to n×100000 for SWV instances. The solution quality TSSA algorithm provides outperforms that of BV in a short time. For example, for SWV06-10 instances, TSSA algorithm offers b-MRE = 6.91% in about three minutes on the personal PC, whereas BV-best provides b-MRE = 8.11% in approximately 180 minutes on the SUN SParc-330. Moreover, for SWV01-10 instances, TSSA algorithm found three new upper bounds, namely: SWV04—1470, SWV08—1756 and SWV10—1754.

Problem	Size	UB(LB)	TSSA			BV[b]			
			Best	M_{av}	T_{av}(s)	SB-RGLS5	CPU(s)	BV-best	CPU(s)
YN1	20×20	885(826)	**884**[d]	891.3	106.3	893	3959.2	891	9382.4
YN2	20×20	909(861)	**907**[d]	911.2	110.4	911	5143.2	910	11647.2
YN3	20×20	892(827)	892	895.5	110.8	897	4016	897	4016
YN4	20×20	968(918)	969	972.6	108.7	977	7407.2	972	10601.2
YN1-4			6.4	6.99	109.1	7.2	5131.4	6.98	8911.7

Table 7. Results for YN1-4 instances

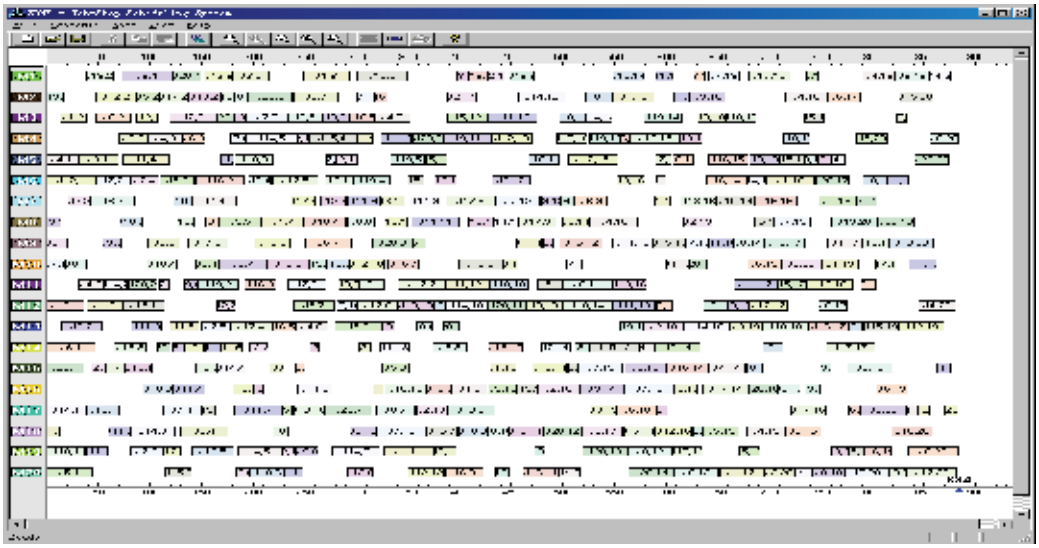


Fig. 10. Gantt chart showing the best solution (makespan is equal to 884) for YN1

			TSSA			BV[b]			
Problem	Size	UB(LB)	Best	M_{av}	T_{av}(s)	SB-RGLS5	CPU(s)	BV-best	CPU(s)
SWV01	20×10	1407	1412	1423.7	142.1	1418	1498	1418	1498
SWV02	20×10	1475	1475*	1480.3	119.7	1484	1389.2	1484	1389.2
SWV03	20×10	1398(1369)	1398	1417.5	139.1	1443	–	1425	3302
SWV04	20×10	1474(1450)	**1470**[d]	1483.7	143.9	1484	1621.2	1483	2433.2
SWV05	20×10	1424	1425	1443.8	146.7	1434	1961.2	1434	1961.2
SWV01-05			0.78	1.76	138.3	1.97	1617.2	1.69	2116.7
SWV06	20×15	1678(1591)	1679	1700.1	192.5	1710	5446	1696	11863
SWV07	20×15	1600(1446)	1603	1631.3	190.2	1645	3903.2	1622	10699
SWV08	20×15	1763(1640)	**1756**[d]	1786.9	190	1787	4264	1785	10375
SWV09	20×15	1661(1604)	1661	1689.2	193.8	1703	4855.2	1672	12151
SWV10	20×15	1767(1631)	**1754**[d]	1783.7	184.6	1794	3005.2	1773	10332
SWV06-10			6.91	8.66	190.2	9.27	4294.7	8.11	11084

Table 8. Results for SWV01-10 instances

7. Conclusion

The efficiency of the tabu search for the JSSP depends on the neighborhood structures and initial solution. In this paper, firstly, a new neighborhood structure is constructed, which could investigate much larger solution space. We compare the new neighborhood structure with the other five neighborhood strategies, and confirm that it is an effective neighborhood structure for the JSSP. Furthermore, the effects of neighborhood evaluation strategies are investigated. Empirical testing discloses that the estimation approach introduced by Balas and Vazacopoulos not only significantly improves the efficiency of the search, but also has no material effect on the solution quality.

Secondly, through the proper use of the structure of the solution space (especially the big valley), we developed the novel hybrid TSSA algorithm which combines the advantage properties of simulated annealing with tabu search strategy. This algorithm mitigates the drawback of tabu search and could reduce the influence of the initial solution and obtain the high-quality solutions in a short running time on a modern PC, which have been confirmed by tests on a large number of benchmark problems. Moreover, it improves a lot of the current best solutions with reasonable computing times. These indicate that TSSA algorithm is a very robust and efficient algorithm for the considered problem. The general idea of the hybrid of tabu search and simulated annealing could also be applied to solving the other difficult combinatorial optimization problems.

In addition, we observed that the TSSA algorithm has better efficiency for the JSSP than the traditional tabu search when $n \leq 2m$; nevertheless, for some rectangle problems ($n >> m$) tabu search with the powerful neighborhood structures and dynamic tabu list could be more effective than the hybrid tabu search approach. Therefore, a subject of future work would exploit the more effective diversification strategy of hybrid tabu search. In addition, how to efficiently set the tabu list in JSSP is still an open problem. The growing researches suggest that the length of short term memory varies dynamically in response to the changing conditions of the search, but there is still a broad research for better implementations.

8. Reference

Adams, J.; Balas, E. & Zawack, D. (1988) The shifting bottleneck procedure for job shop scheduling. Management Science, Vol. 34, pp. 391–401.

Balas E. (1969) Machine sequencing via disjunctive graph: an implicit enumeration algorithm. Operations Research, Vol.17, pp. 941-57.

Balas, E. & Vazacopoulos, A. (1998) Guided local search with shifting bottleneck for job shop scheduling. Management Science, Vol. 44, No. 2, pp. 262–75.

Barnes, J.W.; & Chambers, J.B. (1995) Solving the job shop scheduling problem using tabu search. IIE Transactions, Vol. 27, pp. 257–63.

Blażewicz, J.; Domschke, W. & Pesch, E. (1996) The job shop scheduling problem: Conventional and new solution techniques. European Journal of Operational Research, Vol. 93, pp. 1–33.

Brucker, P. Scheduing algorithm. Berlin: Springer-Verlag, 1995.

Brucker, P.; Jurisch, B. & Sievers, B. (1994) A branch and bound algorithm for job-shop scheduling problem. Discrete Applied Mathematics, Vol. 49, pp. 105–27.

Carlier, J. & Pinson, E. (1989) An algorithm for solving the job shop problem. Management Science, Vol. 35, No.29, pp. 164–76.

Chambers, J.B. & Barnes, J.W. (1996) New tabu search results for the job shop scheduling problem. Technical Report ORP96-10, Graduate Program in Operations Research and Industrial Engineering, The University of Texas at Austin,.

Croce, F.D.; Tadei, R. & Volta, G. (1995) A genetic algorithm for the job shop problem. Computers & Operations Research, Vol. 22, pp. 15–24.

Dell'Amico, M. & Trubian, M. (1993) Applying tabu-search to job-shop scheduling problem. Annals of Operations Research, Vol. 41, No. 1–4, pp.231–52.

Demirkol, E. Mehta, S. & Uzsoy, R. (1998) Benchmarks for shop scheduling problems. European Journal of Operational Research, Vol. 109, pp. 137-41.

Garey, E.L., Johnson, D.S. & Sethi. R. (1976) The complexity of flowshop and job-shop scheduling. Mathematics of Operations Research, Vol. 1, pp. 117–29.

Geyik, F. & Cedimoglu, I.H. (2004) The strategies and parameters of tabu search for job-shop scheduling.Journal of Intelligent Manufacturing, Vol. 15, pp. 439–48.

Giffler, B. & Thompson, G.L. (1960) Algorithms for solving production scheduling problem. Operations Research, Vol. 8, No. 4, pp. 487-503

Glover, F. & Hanafi, S. (2002) Tabu search and finite convergence. Discrete Applied Mathematics, Vol. 119, No. 1-2, pp. 3–36.

Glover, F. & Laguna, M. (1997) Tabu search. Dordrecht: Kluwer Academic Publishers.

Glover, F. (1989) Tabu search—Part I. ORSA Journal on Computing, Vol. 1, No. 3, pp. 190-206.

Glover, F. (1990) Tabu search—Part II. ORSA Journal on Computing, Vol. 2, No. 1, pp. 4-32.

Hanafi, S. (2000) On the convergence of tabu search. Journal of Heuristics, Vol. 7, pp. 47-58.

Hansen P. (1986) The steepest ascent mildest descent heuristic for combinatorial programming. in: Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy.

Jain, A.S. & Meeran, S. (1999) Deterministic job shop scheduling: past, present and future. European Journal of Operational Research, Vol. 113, pp. 390–434.

Jain, A.S.; Rangaswamy, B. & Meeran, S. (2000) New and "stronger" job-shop neighbourhoods: A focus on the method of Nowicki and Smutnicki (1996). Journal of Heuristics, Vol. 6, No.4, pp. 457–80.

Lawrence, S. (1984) Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques (Supplement). Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania,

Matsuo, H.C.; Suh, C.J. & Sullivan, R.S. (1988) A controlledsearch simulatedannealing methodfor general job shop scheduling problem. Working Paper, 03-04-88, Department of Management, The University of Texas at Austin, Austin, TX.

Muth, J.F. & Thompson, G.L. (1963) Industrial scheduling. Englewood Cliffs, NJ: Prentice Hall.

Nowicki E, Smutnicki C. (2002) Some new tools to solve the job shop problem. Technical Report 60/2002, Institute of Engineering Cybernetics, Technical University of Wroclaw, Wroclaw, Poland.

Nowicki, E. & Smutnicki, C. (1996) A fast taboo search algorithm for the job shop scheduling problem. Management Science, Vol. 42, No. 6, pp. 797–813.

Nowicki, E. & Smutnicki, C. (2001) Some new ideas in TS for job-shop scheduling. Technical Report 50/2001, Institute of Engineering Cybernetics, Wroclaw University of Technology, Wroclaw, Poland.

Nowicki, E. & Smutnicki, C. (2002) Some new tools to solve the job shop problem. Technical Report 60/2002, Institute of Engineering Cybernetics, Technical University of Wroclaw, Wroclaw, Poland.

Pezzella, F. & Merelli, E. (2000) A tabu search method guided by shifting bottleneck for job shop scheduling problem. European Journal of Operational Research, Vol. 120, pp. 297–310.

Storer, R.H.; Wu, S.D. & Vaccari, R. (1992) New search spaces for sequencing problems with applications to job-shop scheduling. Management Science, Vol. 38, No. 10, pp. 1495-1509.

Taillard, É.D. (1989) Parallel taboo search technique for the job shop scheduling problem. Technical Report ORWP 89/11, DMA, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland.

Taillard, É.D. (1994) Parallel taboo search techniques for the job-shop scheduling problem. ORSA Journal on Computing, Vol. 6, pp. 108–17.

Ten Eikelder, H.M.M.; Aarts, B.J.M; Verhoeven, M.G.A. & Aarts, E.H.L. (1999) Sequential and paralled local search algorithms for job shop scheduling. In: Voss S, Martello S, Osman IH, Roucairol C, editors. Meta-Heuristic:advances and Trends in Local Search Paradigms for Optimization. Massachusetts: Kluwer Academic Publishers, pp. 359–71.

Vaessens, R.J.M.; Aarts, E.H.L. & Lenstra, J.K. (1996) Job shop scheduling by local search. INFORMS Journal on Computing, Vol. 8, pp. 302–17.

Van Laarhoven, P.J.M.; Aarts, E.H.L. & Lenstra, J.K. (1992) Job shop scheduling by simulated annealing. Operations Research, Vol. 40 No. 1, pp. 113–25.

Yamada, T. & Nakano, R. (1992) A genetic algorithm applicable to large-scale job-shop problems. in: Manner R, Manderick B (Eds.). Proceedings of the Second International Workshop on Parallel Problem Solving from Nature (PPSN'2), Brussels, Belgium, pp.281-290.

Part II:

Tabu Search Applications

9

Tabu Search Experience in Forest Management and Planning

Dr. Pete Bettinger
University of Georgia, Athens
USA

1. Introduction

Tabu search was initially developed by Glover (1989, 1990), and has been applied to a number of forest management and planning problems (Murray & Church 1995; Bettinger et al. 1997, 1998, 2002; Boston & Bettinger 1999; Brumelle et al. 1998). In general, when using tabu search to address forest management and planning problems, a number of forest plans are deterministically developed and assessed, each subsequent plan being slightly different than its predecessor, and thus each is considered an iteration of the modeling process. A large number of iterations are usually required to ensure that the search process has explored the solution space sufficiently. In most cases in forest planning, tabu search is used as a 1-opt search process, where a feasible forest plan is modified by changing the status (harvest timing, prescription, etc.) of a single forest management unit, thus creating a new plan. However, as we will see, other intensification and diversification processes have been used to expand the capabilities of the search process.

A tabu search process begins with an initial, randomly defined, feasible solution (forest plan) (Figure 1). A simple Monte Carlo (i.e., random) process is generally used to select timber stands and management prescriptions, and constraints are assessed with programming logic to ensure that each choice results in a feasible solution. Feasibility is not difficult to obtain in the initial solution, as most choices are made by avoiding the violation of constraints. However, the initial solution is generally of low quality. This process is consistent with much of the work related to the use of heuristics in forestry (e.g., Bettinger et al. 1998). With each iteration (k) of the tabu search algorithm, a new feasible solution (x_k) is created from a transformation of the previous feasible solution (x_{k-1}) by a move (δ). A δ is a transition from one feasible solution to another feasible solution. The δ may represent the change that results in the best possible improvement in solution x_{k-1}, or that results in the least deterioration in the value of x_{k-1} (Voβ 1993).

With this search technique, a δ can consist of assigning a different prescription to a timber stand (1-opt δ) or swapping the prescriptions assigned to two different timber stands (2-opt δ). A candidate δ cannot consist of the assignment of more than one prescription to a timber stand. Each feasible δ requires that it does not result in a violation of the constraints. In all cases, a tabu tenure is assigned to each δ and aspiration criteria are employed. Most forest planning applications of tabu search involve the scheduling of harvests to timber stands. However, the allocation of timber stands and cutting patterns to logging systems has been

explored with tabu search (Murphy 1998), as have optimal methods for bucking (cutting) logs from trees (Laroze & Greber 1997).

2. Intensification processes within tabu search

Recently, some have suggested that enhancements are necessary to tabu search to allow it to locate increasingly efficient solutions for complex forest planning problems, such as those facing forest landowners (Bettinger et al. 1999, Caro et al. 2003, Richards & Gunn 2000). One enhancement to tabu search that has shown promise in forest management applications is the addition of a 2-opt search strategy. A 1-opt search strategy in forest management planning is generally used to select the management prescription or harvest timing for a stand of trees. Simply stated, the best change to a forest plan is selected from the tabu search neighborhood, and if not tabu (or if tabu and if it passes the aspiration criteria test), the change is incorporated into the forest plan. A 2-opt search strategy is used to switch the prescription or timing of harvest of two separate stands of trees. This switching results in less of an impact on the objective function, and results in changes to forest plans that might not otherwise be acceptable in similar consecutive 1-opt choices. Several studies in the forest management literature have confirmed the notion that the addition of this process to a tabu search procedure will lead to significantly higher quality forest plans (Bettinger et al. 1999, Boston & Bettinger 2001a, Bettinger et al. 2002, Caro et al. 2003, Batten et al. 2005).

There are two drawbacks to the use of the 2-opt process in forest planning. First, it cannot be used in the absence of a 1-opt process. The 2-opt process can only switch management opportunities that are currently in a forest plan. For example, if all of the timber stands in a forest plan are scheduled for harvest in the first year of a ten-year plan, there is no opportunity with the 2-opt process to schedule stands in the other nine years. A 1-opt process provides these opportunities by changing the harvest timing of individual timber stands. Second, the 2-opt process is an intensive process from a computer processing standpoint, since the full neighborhood is about 0.5 ($n \times n$) in size rather than ($n \times p$), where n represents the number of stands, and p represents the number of prescriptions or harvest timing choices (generally a small number). This is important when developing a tactical plan that may cover 20 or more years and involve thousands of timber stands. Caro et al. (2003) investigated the use of 2-opt neighborhoods as well, yet in this case, the 2-opt process was only used to mitigate infeasibilities generated by 1-opt moves.

3. Diversification processes within tabu search

A number of diversification processes can be used to force a tabu search process to un-explored areas of the solution space. One involves the use of long-term memory. Here, the frequency with which choices are evaluated for each stand of trees is tracked, and at some point during the search, the least-selected choices can be forced into the solution. Boston & Bettinger (1999) illustrated the use of such a diversification process in a forest planning problem, however they found that simulated annealing was able to produce better solutions, even though tabu search produced solutions with less variability. Brumelle et al. (1998) tested frequency-based diversification within tabu search to force the process to test harvesting alternatives that were rarely selected during the development of a forest plan. In this work, a static diversification scheme was used to determine when to diversify the

search. This consisted of initiating the process every x iterations, the size of which was explored through trial-and-error. In addition, a process for determining when to diversify was developed whereby the diversification process was initiated whenever the average improvement in objective function value fell below some pre-defined threshold. Neither of these diversification schemes, once optimal parameters were identified, outperformed the other.

Pukkala & Heinonen (2006) suggest that when applying tabu search to forest management problems, one way to prevent the increase in computational time as a problem gets larger is to decrease the size of the neighborhood. In a large forest planning problem, Bettinger et al (2007) described how a region-limited neighborhood could be used in conjunction with tabu search to effectively allow the development of a forest plan. The management problem involved scheduling stand-density management prescriptions to over 17,000 timber stands contained within a 178,000 ha watershed in eastern Oregon (USA). Two types of tabu search were utilized: 1-opt moves only (changing the prescription of a single timber stand), and 1-opt and 2-opt moves (switching the prescriptions of two timber stands). The objective of the problem was to locate a plan that provided the highest, and most even timber harvest volume over a 100-year time horizon. The attainment of the objective was complex in that the prescriptions available to each timber stand involved partial cutting activities that produced timber volume at various points in time through the 100-year time period. Given the size of the problem and the observation that developing a full neighborhood for 1-opt moves was computationally slow, the search was adjusted to examine 1-opt moves only within a 2,000 timber stand window. This window shifted 1 stand upward with each iteration of the tabu search process. When used, the 2-opt neighborhood was even smaller (1,000 stands), and shifted by 20 stands with each iteration of the search process. While using the full neighborhood produced harvest levels that were, on average, consistent with the region-limited process, the region-limited process produced harvest levels that contained less variation across the 100-year time horizon, thus these solutions were noted as being superior to those derived when using the full neighborhood. However, some of this improvement is attributable to the use of the 2-opt neighborhood of choices. Caro et al. (2003) also used sub-region tabu neighborhoods to limit the amount of computations necessary for each iteration of tabu search. And Heinonen & Pukkala (2004) diversified their tabu search process by developing neighborhoods of randomly-chosen 1-opt and 2-opt moves. In this sense, a full neighborhood was not developed due to the size of the problem, but a smaller neighborhood composed of randomly chosen moves was employed.

Penalty functions have also been incorporated into tabu search processes to allow the search to more freely explore the solution space. Caro et al. (2003) allowed infeasible 1-opt moves to be accepted during a search, yet these infeasibilities were either mitigated using 2-opt moves, or a penalty was added to the objective function value to force (hopefully) the infeasibility to eventually go away. Richards & Gunn (2003) describe a strategic oscillation process where infeasible 1-opt moves are acceptable, yet they were penalized in the objective function using a biased penalty weighting procedure. Here, solutions (forest plans) that were "near" the boundary of the feasible region were penalized less than solutions that had moved further away from the boundary. The conclusion from these efforts is that for some problems, allowing solutions to temporarily deviate from feasibility could result in higher quality solutions.

4. Tabu tenure adjustment

In general, when applied to forest planning problems, the tabu tenure is generally fixed. To locate the best fixed tabu tenure, this requires an extensive examination of a range of tabu tenures to locate those that produce high quality solutions. Not only is this parameterization process time-consuming, but also it must be performed for *each forest planning problem*. The size (number of timber stands and number of management actions that can be applied to each timber stand) and the complexity of each forest planning problem are so diverse that this situation precludes the attainment of a single tabu tenure rule for all problems. However, Heinonen & Pukkala (2004) suggested that the use of a tabu tenure which was one-fifth of the potential size of the tabu list might be appropriate in some instances. Modifications to a fixed tabu tenure have been explored in several forest planning efforts. For example, a randomly determined tabu tenure was described in Caro et al. (2003), Brumelle et al. (1998), and Pukkala & Kurttila (2005). This process randomly determines the length of the tabu tenure for each choice that was made, the length of which is usually confined to a suitable range of tabu tenures.

Richards & Gunn (2003) suggested the use of a reactive tabu tenure, where when cycling of solutions is recognized, the tabu tenure is increased, and when cycling of solutions is not apparent, the tabu tenure is decreased. However, through partial experimentation of this method, they concluded that a fixed tabu tenure would be more appropriate, and that the extensive trial-and-error experimentation of tabu search to determine a tabu tenure could not be avoided. Pukkala & Kurtilla (2005) went further to assess the optimal range of the tabu tenure, and their work provides us with some guidance for certain types of forest planning problems.

5. Meta-models involving tabu search

A number of forest planning meta-models involving tabu search have been described in the literature. For example, tabu search has been used in conjunction with linear programming to illustrate how strategic and tactical goals could be met within a hierarchical system. In Boston & Bettinger (2001b), linear programming was used to solve a long-term forest planning problem, which was considered "relaxed" since spatial constraints were not incorporated into the planning system. Tabu search then used the outcomes of the optimal plan (harvest levels) to develop a tactical plan where spatial considerations were recognized and controlled. In this system the objective was to minimize the deviations between the harvest levels suggested by linear programming, and the harvest levels that could be accommodated in a tactical plan.

Within forest management planning, tabu search has also been used in conjunction with several other heuristics in an effort to capitalize on the search behavior of each. For example, Boston & Bettinger (2001a) combine forest plans developed with tabu search with a genetic algorithm process where the better tabu search plans are periodically split and recombined at a randomly defined crossover point to diversify the search. In another area of work, Boston & Bettinger (2002) compared two meta heuristics that involved tabu search: (1) a 1-opt tabu search process and a genetic algorithm process, and (2) a 1-opt and a 2-opt tabu search process and a genetic algorithm process. Here, the latter of the two provided the better solutions for the planning problems because, as we noted earlier, the addition of a 2-opt process generally improves the quality of the outcomes. Li (2007) recently explored

numerous combinations of tabu search, simulated annealing, threshold accepting, and the raindrop method (Bettinger & Zhu 2006) to determine whether a meta-model could be developed that would capitalize on the search behavior of each. The transition between search processes in Li (2007) was determined by continuously assessing the quality of solutions, thus acquiring knowledge of the behavior, then switching heuristic processes when further improvements in solution quality were seemingly lacking. Nalle et al. (2004) embedded the tabu search neighborhood structure within a simulated annealing search process. Here, a neighborhood was defined as a forested stand and its adjacent neighbors, and when stand was selected for a 1-opt move, after the move was incorporated into the solution, its neighbors were then randomly selected for 1-opt moves using a simulated annealing process for a small number of iterations. Afterwards, another forested stand was selected, and its neighbors subjected to a simulated annealing process, and so on. This integration of techniques allowed small areas of the solution space to be more thoroughly analyzed than larger areas, and interestingly, the combination of techniques was computationally faster than the two methods working alone on the same problem.

6. Incorporation of non-linear goal assessment processes

One of the most common non-linear goal assessments in contemporary tactical forest planning processes involves the timing and placement of clearcut harvesting activities. The maximum size of clearcut harvests has been noted in several laws and policies related to the management of forests in North America. For example, in the United States, some state laws, such as those in place in Oregon, California, and Washington, recognize clearcut size limitations (Boston and Bettinger 2006). Some U.S. National Forests have limits on the sizes of clearcuts, such as the 16 ha maximum on the Chattahoochee-Oconee National Forest (U.S. Department of Agriculture, Forest Service 2004). Many industrial forestry organizations in the southeastern United States have also adopted the Sustainable Forestry Initiative (Sustainable Forestry Initiative, Inc. 2005) to show a commitment to social responsibility, and to demonstrate that their forests are managed in a sustainable manner. Like the Forest Stewardship Council (Forest Stewardship Council - U.S. 1996), participation in the Sustainable Forestry Initiative program is voluntary, and each program contains a number of principles and objectives that need to be implemented and achieved. One of the performance measures in the Sustainable Forestry Initiative program relates to the size, shape, and placement of clearcut harvests, which restricts the average size of clearcuts 48 ha or less. In addition, many companies have developed internal policies to voluntarily limit the maximum clearcut sizes to 96 ha or less (Boston and Bettinger 2001a). Some private landowners have also expressed an interest in adhering to these principles and objectives without the formality of becoming a member of a certification program (Batten et al. 2005).

Controlling where and when harvests are placed on a landscape requires an additional, perhaps extensive, set of non-linear constraints to a forest planning problem formulation. There are two conceptual models of adjacency and green-up commonly used in forest planning, the unit restriction model (URM) and the area restriction model (ARM), both of which are described in detail by Murray (1999). The URM controls the placement of activities by precluding the scheduling of one harvest that might touch (or be near) another harvest that has already been scheduled. The ARM controls the size of activities being scheduled by allowing adjacent activities to be scheduled concurrently (within the green-up period) as long as the total size does not exceed some pre-defined maximum. If timber

stands are small relative to the maximum harvest area, URM constraints may significantly mis-represent the problem (Barrett and Gilless 2000), a concern that must be kept in mind, since changing the maximum harvest area in a planning problem is relatively easy, while changing the average size of stands of trees maintained in a geographic information system is relatively difficult. The ARM is not constrained in this manner, and can rely on the most disaggregate data available (Murray and Weintraub 2002). In our assessment of the recent literature, the URM and ARM are equally addressed, suggesting that each method has value even though they have their limitations. Inherent in each conceptual model is the green-up period, or exclusion, period. This is the length of time that must pass before activities are allowed in adjacent management units or stands. The green-up period is usually expressed in terms of years (2-4 years in the southern United States, for example, and 5-20 years in the western United States). Conceptually, it represents the amount of time a regenerated stand needs to "green-up," or grow to a certain height, before an adjacent stand is allowed to be harvested.

Area-based (ARM) clearcut size restrictions can require a large number of constraints to enable the control of adjacent harvests. As the potential maximum clearcut size increases relative to the average size of a management unit or stand, the number of constraints and potential constraint redundancies increases (Yoshimoto and Brodie 1994, Crowe et al. 2003). A number of methods have been assessed for generating sets of constraints that will increase the efficiency of integer programming problem solving methods, since a reduction in constraints plays an important role in the amount of time needed to solve a problem. Yoshimoto & Brodie (1994) and Murray & Church (1995) describe several constraint constructs that can be used to represent adjacency relationships in forest planning problems. The formulation of adjacency constraints has taken on considerable debate over the last 20 years, with the early work by Jones et al. (1991), who suggested that different formulations of constraints needed to be tested to determine their efficiency in assisting with the solving of a problem. Later work by Yoshimoto & Brodie (1994), Murray & Church (1995), McDill et al. (2002), and Goycoolea et al. (2005), to name a few, proceeded to do just that - examine how adjacency constraints might be arranged to solve a problem exactly, and quickly. Along these lines, constraints are either simply combined to reduce redundancy, or developed for cliques of stands, or developed for sets of mutually adjacent units (Murray & Church 1996). While the rather simplistic pair-wise adjacency constraints can be used for URM problems, they may not provide the most efficient formulation for integer programming problems (Murray & Church 1995). However, these types of constraints are commonly used in heuristic methods because they can quickly be assessed (Bettinger et al. 2002). Pair-wise adjacency constraints typically take the form of:

$$S1P1 + S2P1 \leq 1$$

Where S1P1 is a binary integer that represents the potential harvest of timber stand 1 during time period 1, and S2P1 is a binary integer that represents the potential harvest of timber stand 2 during time period 1. If these timber stands are considered adjacent, the constraint prevents them from being harvested in the same time period.

URM adjacency constraints are evaluated by assessing the status of all adjacent timber stands to each timber stand that is being considered for clearcut harvest. ARM constraints are evaluated, as previously noted, using logic to check all neighbors of clearcut timber stand *n*, and if they are clearcut within the green-up window, their neighbors, and so on

until the sprawling cluster has been fully identified. The two conceptual models of adjacency are not mutually exclusive, and there are times when a planning effort may require both ARM and URM methods. For example, the ARM constraints could be used to control the size of a harvested area (forming a harvest block), while the URM constraints could be used to prevent two harvest blocks from merging together (Bettinger & Johnson 2003, Bettinger et al. 2005). In addition, the ARM method need not be simply used to control the maximum harvest area. There may be management instances where the average harvest opening size is more important than the maximum opening size (Boston & Bettinger 2001b). There are two generally methods for acknowledging these constraints within tabu search: (1) assess the constraints during the development of the tabu search neighborhood, and (2) assess the constraints after a choice has been selected from the tabu search neighborhood. The former approach can be computationally expensive, particularly when a full neighborhood is being used, and when ARM constraints are being assessed. However, the latter approach may also be computationally wasteful, since the choices made from the neighborhood are not necessarily feasible until the constraints have been assessed.

One of the first tabu search papers in forest management that involved non-linear wildlife habitat goals was presented by Bettinger et al. (1997). Here, a standard 1-opt tabu search process was used to maximize an even-flow timber harvest objective while also meeting constraint levels for the development and maintenance of Rocky Mountain elk (*Cervus elaphus nelsoni*) habitat. In this work, 80% of the elk forage area, which included land that was recently cleared or contained timber stands less than 10 years old had to be within 200 m of elk hiding cover, which was composed of timber stands containing trees at least 40 years old. The hiding cover areas also had to be at least 3 ha in size. In addition, 80% of the elk forage area had to be within 300 m of elk hiding cover, which was composed of timber stands at least 40 years old, and when aggregated, were at least 17 ha in size. The data describing the landscape was in grid form, and a moving window assessment was performed subsequent to the choice selected from the neighborhood. If after performing the moving window analysis the constraints were violated, then the potential choice was unscheduled. This work illustrated the search behavior of tabu search in the development of a forest plan, and suggested that three phases of the search were evident: (a) a hill-climbing phase where improvements to the objective were found, (b) an adjustment phase composed of both improvements and declines in objective function quality, and (c) a steady-state phase where no more improvements were recognizable. When further constrained, the steady-state phase was no longer evident, and was replaced by a heavily constrained phase, where general declines in objective function quality were noted after the best solution was located in the adjustment phase. More than likely, this was due to the fixed tabu tenure that was assumed.

Bettinger et al. (1998) later developed a tabu search process for scheduling timber harvests subject to stream sediment and stream temperature constraints. Here, after the scheduling of each timber harvest an assessment of sediment and temperature levels in the stream system was made. These assessments used procedures that involved logic and rules of thumb to assess aquatic habitat quality, which were distinctly non-linear in nature. Further, when sediment levels exceeded maximum threshold levels, three options were available to the scheduling model: (1) unschedule the timber harvest, (2) change the status of a road, perhaps decommissioning one, or (3) assign to a road the use of central tire inflation in all logging trucks (reduced air pressure levels in the tires). Each of these in effect can reduce

sediment levels in the stream system. In developing this scheduling process, two networks were necessary. The first involved the stream system. All of the sediment and temperature effects within the watershed were routed down the stream system to the exit point of the watershed. It was at this exit point that sediment and temperature levels were constrained. This process was consistent with the acquisition of real data from field gauging stations, and provided a validation of the aquatic assessments. The second network involved the road system. Here, access to all scheduled timber harvests needed to be maintained. If a road was scheduled for decommissioning, alternative access routes needed to be available. A shortest path algorithm was used to ensure that the schedule harvests could be transported out of the watershed at all times during the development of the forest plan. This work by Bettinger et al. (1998) was also the first in the forest sciences to utilize extreme value theory in estimating the global optimum solution to a forest planning problem. Since the aquatic assessments were distinctly non-linear in nature, and given the size of the problem, an exact solution was unobtainable.

7. Conclusions

In our experiences with tabu search in forest planning efforts, we have learned that the basic 1-opt scheduling process produces relatively good results to difficult problems. However, the incorporation of intensification processes, such as a 2-opt neighborhood of choices, always leads to higher quality solutions. However, using a 2-opt search neighborhood is computationally expensive, and therefore increases the time required to obtain a solution. Diversification processes, such as the use of strategic oscillation, can also force the search to un-explored areas of the solution space, and can result in higher quality solutions as well. A region-limited approach for the development of tabu search neighborhoods can also be of value in both diversifying the search as well as reducing the time required to generate a high-quality solution. Determining the appropriate tabu tenure to use in forest planning problems is an area of work that needs further exploration. Almost every forest planning problem requires a different tabu tenure assumption, given differences in problem size and complexity. Some approaches for varying the tabu tenure, and for assuming that the appropriate tenure is a function of the size of the problem, have both been explored, however no firm assumption has been proposed. Tabu search is a process that is adaptable to the various non-linear functional relationships that are becoming common in forest plans. As with other heuristics, the advantage of this search process over traditional mathematical programming methods (e.g., mixed integer programming) is that it allows forest planners to more fully capture the essence of a planning situation, and allows us to thereby develop forest plans that may not require further assessment prior to implementation.

8. References

Barrett, T.M., and J.K. Gilless. 2000. Even-aged restrictions with sub-graph adjacency. Annals of Operations Research. 95: 159-175.

Batten, W.H. Jr., P. Bettinger, and J. Zhu. 2005. The effects of spatial harvest scheduling constraints on the value of a medium-sized forest holding in the southeastern United States. Southern Journal of Applied Forestry. 29: 185-193.

Bettinger, P., K. Boston, Y.-H. Kim, and J. Zhu. 2007. Landscape-level optimization using tabu search and stand density-related forest management prescriptions. European Journal of Operational Research. 176: 1265-1282.

Bettinger, P., K. Boston, and J. Sessions. 1999. Intensifying a heuristic forest harvest scheduling search procedure with 2-opt decision choices. Canadian Journal of Forest Research. 29: 1784-1792.

Bettinger, P., D. Graetz, K. Boston, J. Sessions, and W. Chung. 2002. Eight heuristic planning techniques applied to three increasingly difficult wildlife planning problems. Silva Fennica. 36: 561-584.

Bettinger, P., and K.N. Johnson. 2003. Spatial scheduling of forest management activities using a dynamic deterministic harvest block aggregation process. Journal of Forest Planning. 9: 25-34.

Bettinger, P., M. Lennette, K.N. Johnson, and T.A. Spies. 2005. A hierarchical spatial framework for forest landscape planning. Ecological Modelling. 182: 25-48.

Bettinger, P., J. Sessions, and K. Boston. 1997. Using Tabu search to schedule timber harvests subject to spatial wildlife goals for big game. Ecological Modelling. 94: 111-123.

Bettinger, P., J. Sessions, and K.N. Johnson. 1998. Ensuring the compatibility of aquatic habitat and commodity production goals in eastern Oregon with a Tabu search procedure. Forest Science. 44: 96-112.

Bettinger, P., and J. Zhu. 2006. A new heuristic for solving spatially constrained forest planning problems based on mitigation of infeasibilities radiating outward from a forced choice. Silva Fennica. 40: 315-333.

Boston, K., and P. Bettinger. 1999. An analysis of Monte Carlo integer programming, simulated annealing, and tabu search heuristics for solving spatial harvest scheduling problems. Forest Science. 45: 292-301.

Boston, K., and P. Bettinger. 2001a. The economic impact of green-up constraints in the southeastern United States. Forest Ecology and Management. 145: 191-202.

Boston, K., and P. Bettinger. 2001b. Development of spatially feasible forest plans: A comparison of two modeling approaches. Silva Fennica. 35: 425-435.

Boston, K., and P. Bettinger. 2002. Combining tabu search and genetic algorithm heuristic techniques to solve spatial harvest scheduling problems. Forest Science. 48:35-46.

Boston, K., and P. Bettinger. 2006. An economic and landscape evaluation of the green-up rules for California, Oregon, and Washington (USA). Forest Policy and Economics. 8: 251-266.

Brumelle, S., D. Granot, M. Halme, and I. Vertinsky. 1998. A tabu search algorithm for finding good forest harvest schedules satisfying green-up constraints. European Journal of Operational Research. 106: 408-424.

Caro, F., M. Constantino, I. Martins, and A. Weintraub. 2003. A 2-opt tabu search procedure for the multiperiod forest harvest scheduling problem with adjacency, greenup, old growth, and even flow constraints. Forest Science. 49: 738-751.

Crowe, K., J. Nelson, and M. Boyland. 2003. Solving the area-restricted harvest scheduling model with the branch and bound algorithm. Canadian Journal of Forest Research. 33: 1804-1814.

Forest Stewardship Council - U.S., 1996. Principles and criteria. Forest Stewardship Council - U.S., Washington, D.C. http://www.fscus.org/standards_criteria/ (Accessed 6/26/07).

Glover, F. 1989. Tabu search - Part I. ORSA Journal on Computing. 1: 190-206.

Glover, F. 1990. Tabu search - Part II. ORSA Journal on Computing. 2: 4-32.

Goycoolea, M., A.T. Murray, F. Barahona, R. Epstein, and A. Weintraub. 2005. Harvest scheduling subject to maximum area restrictions: Exploring exact approaches. Operations Research. 53: 490-500.

Heinonen, T., and T. Pukkala. 2004. A comparison of one- and two-compartment neighbourhoods in heuristic search with spatial forest management goals. Silva Fennica. 38: 319-332.

Jones, J.G., B.J. Meneghin, and M.W. Kirby. 1991. Formulating adjacency constraints in linear optimization models for scheduling projects in tactical planning. Forest Science. 37: 1283-1297.

Laroze, A.J., and B. Greber. 1997. Using tabu search to generate stand- level, rule-based bucking patterns. Forest Science. 43: 157-169.

Li, R. 2007. Integration of GIS techniques and heuristic algorithms to address spatial forest planning issues in the southern U.S. PhD. Dissertation, University of Georgia, Athens. 159 p.

McDill, M.E., S.A. Rebain, and J. Braze. 2002. Harvest scheduling with area-based adjacency constraints. Forest Science. 48: 631-642.

Murphy, G. 1998. Allocation of stands and cutting patterns to logging crews using a tabu search heuristic. Journal of Forest Engineering. 9: 31-37.

Murray, A.T. 1999. Spatial restrictions in harvest scheduling. Forest Science. 45: 45-52.

Murray, A.T., and R.L. Church. 1995. Measuring the efficacy of adjacency constraint structure in forest planning models. Canadian Journal of Forest Research. 25: 1416-1424.

Murray, A.T., and R.L. Church. 1996. Analyzing cliques for imposing adjacency restrictions in forest models. Forest Science. 42: 166-175.

Murray, A.T., and A. Weintraub. 2002. Scale and unit specification influences in harvest scheduling with maximum area restrictions. Forest Science. 48: 779-789.

Nalle, D.J., C.A. Montgomery, J.L. Arthur, S. Polansky, and N.H. Schumaker. 2004. Modeling joint production of wildlife and timber. Journal of Environmental Economics and Management. 48: 997-1017.

Pukkala, T., and T. Heinonen. 2006. Optimizing heuristic search in forest planning. Nonlinear Analysis: Real World Applications. 7: 1284-1297.

Richards, E.W., and E.A. Gunn. 2000. A model and tabu search method to optimize stand harvest and road construction schedules. Forest Science. 46: 188-203.

Richards, E.W., and E.A. Gunn. 2003. Tabu search design for difficult forest management optimization problems. Canadian Journal of Forest Research. 33: 1126-1133.

Sustainable Forestry Initiative, Inc. 2005. Sustainable Forestry Initiative® Standard (SFIS), 2005-2009 Standard. Sustainable Forestry Initiative, Inc., Arlington, VA. 28 p. http://www.sfiprogram.org/generalPDFs/SFBStandard2005-2009.pdf (accessed 6/26/07).

U.S. Department of Agriculture, Forest Service. 2004. Land and Resource Management Plan, Chattahoochee-Oconee National Forests. U.S. Department of Agriculture, Forest Service, Southern Region, Atlanta, GA. Management Bulletin R8-MB 113 A.

Voβ, S. 1993. Tabu search: Applications and prospects. *In* Network Optimization Problems, Zhu, D.Z., and P.M. Pardalos (eds.). World Scientific Publishing Co., Singapore. pp. 333-353.

Yoshimoto, A., and J.D. Brodie. 1994. Comparative analysis of algorithms to generate adjacency constraints. Canadian Journal of Forest Research. 24: 1277-1288.

10

Feature Selection using Intensified Tabu Search for Supervised Classification

Muhammad Atif Tahir and Jim Smith
Department of Computer Science, Bristol Institute of Technology,
University of the West of England,
UK

1. Introduction

Feature selection algorithms are popular methods to reduce the dimensionality of the feature space and remove the redundant, irrelevant or noisy data. The term feature selection refers to the selection of the best subset of the input feature set. These methods used in the design of pattern classifiers have three goals:

1. to reduce the cost of extracting the features
2. to improve the classification accuracy
3. to improve the reliability of the estimation of the performance, since a reduced feature set requires less training samples in the training process of a pattern classifier [1, 2]

Feature selection produces savings in the measuring features (since some of the features are discarded) and the selected features retain their original physical interpretation [1]. This feature selection problem can be viewed as a multiobjective optimisation problem since it involves minimising the feature subset while maximizing the classification accuracy.

Mathematically, the feature selection problem can be formulated as follows. Suppose X is an original feature vector with cardinality n and $\overline{X}$ is the new feature vector with cardinality $\overline{n}$, $\overline{X} \subseteq X$, $J(\overline{X})$ is the selection criterion function for the new feature vector $\overline{X}$. The goal is to optimize J().

Feature selection problem is NP-hard (Non-deterministic Polynomial-time hard) [3, 4]. Therefore, the optimal solution can only be achieved by performing an exhaustive search in the solution space [5]. However, exhaustive search is feasible only for small n where n is the number of features. A number of algorithms have been proposed for feature selection to obtain near-optimal solutions [1, 2, 6, 7, 8, 9, 10, 30]. The choice of an algorithm for selecting the features from an initial set depends on n. The feature selection problem is said to be of small scale, medium scale, or large scale according to n belonging to the intervals [0,19], [20,49], or [50,1], respectively [2, 8]. Sequential Forward Selection (SFS) [11] is the simplest greedy sequential search algorithm and has been used for land mine detection using multispectral images [12]. Other sequential algorithms such as Sequential Forward Floating Search (SFFS) and Sequential Backward Floating Search (SBFS) are more efficient than SFS and usually find fairly good solutions for small and medium scale problems [7]. However, these algorithms suffer from the deficiency of converging to local optimal solutions for large

scale problems when $n > 50$ [2, 8]. Recent iterative heuristics such as tabu search and genetic algorithms have proved to be effective in tackling this category of problems which are characterized by having an exponential and noisy search space with numerous local optima [8, 9, 13, 14].

Feature selection algorithms can be broadly divided into two categories [32, 31]: *filters*, and *wrappers*. The filter approach evaluates the relevance of each feature or feature subset using the data set alone and without using any machine learning algorithm [32]. On the other hand, Wrapper approach uses machine learning algorithms to evaluate the relevance of feature subset [31]. An extensive summary of different filter and wrapper approaches to feature selection is provided by [33]. As reported in [31], when the objective is to maximize the classification accuracy of a given feature subset, the features selected should depend not only on relevance of the data but also on the machine learning algorithm. For that reason, throughout this chapter, the wrapper feature selection approach using Naive Bayes classifier has been adopted.

In previous papers [23, 24, 25], we have proposed tabu search (TS) based computational intelligence techniques to solve feature selection problems. Nearest Neighbor classifier was used previously as objective function. In this chapter; instead of instance based classifier; we will explore Naive Bayes Classifier compared with sequential feature selection algorithms. The aim is to maximize the classification accuracy while minimizing the number of features.

This chapter is organized as follows. Section 2 gives an overview about Tabu Search and Fuzzy objective function proposed in [25] followed by fea ture selection using Tabu Search proposed originally by [8] and then modified by Tahir et al [27], by introducing feature search intensification strategy in Section 3. Section 4 discusses experiments carried out followed by discussion on other advanced Tabu Search approaches for classification problems in sec- tion 5. Section 6 concludes the chapter.

2. Overview of Tabu Search, and fuzzy logic

TS was introduced by Fred Glover [15, 16] as a general iterative metaheuristic for solving combinatorial optimisation problems. Tabu Search is conceptually simple and elegant. It is a form of local neighbourhood search. Each solution $S \in \Omega$ has an associated set of neighbours $N(S) \subseteq \Omega$ where Ω is the set of feasible solutions. A solution $S' \in N(S)$ can be reached from S by an operation called a *move to* S'. TS moves from a solution to its best admissible neighbour, even if this causes the objective function to deteriorate. To avoid cycling, solutions that were recently explored are declared forbidden or tabu for a number of iterations. The tabu status of a solution is overridden when a certain criteria (aspiration criteria) are satisfied. Sometimes intensification and diversification strategies are used to improve the search. In the first case, the search is accentuated in the promising regions of the feasible domain. In the second case, an attempt is made to consider solutions in a broad area of the search space. The Tabu Search algorithm is given in Algorithm 1.

The size of tabu list can be determined by experimental runs, watching for the occurrence of cycling when the size is too small, and the deterioration of solution quality when the size is too large [?]. Suggested values of tabu list size include Y; $\sqrt{Y}$ (where Y is related to problem size, e.g. number of modules to be assigned in the quadratic assignment problem (QAP), or the number of cities to be visited in the travel salesman problem (TSP), and so on) [13].

Algorithm 1 Algorithm Tabu Search (TS)

Ω : **Set of feasible solutions**
S : Current Solution
S^* : Best admissible solution
$Cost$: Objective function
$N(S)$: Neighbourhood of solution S
V^* : Sample of neighbourhood solutions
T : Tabu list
AL : Aspiration Level

Begin
Start with an initial feasible solution $S \in \Omega$.
Initialize tabu list and aspiration level.
For fixed number of iterations Do
    Generate neighbour solutions $V^* \subset N(S)$.
    Find best $S^* \in V^*$.
    If move S to S^* is not in T Then
        Accept move and update best solution.
        Update tabu list and aspiration level.
        Increment iteration number.
    Else
        If $Cost(S^*) < AL$ Then
            Accept move and update best solution.
            Update tabu list and aspiration level.
            Increment iteration number.
        End If
    End If
End For
End

2.1 Fuzzy logic

In this chapter, we present an intermediate TS algorithm, where the quality of a solution is characterized by a fuzzy logic rule expressed in linguistic variables of the problem domain. Fuzzy set theory has recently been applied in many areas of science and engineering. In the most practical situations, one is faced with several concurrent objectives. Classic approaches usually deal with such difficulty by computing a single utility function as a weighted sum of the individual objectives, where more important objectives are assigned higher weights [25]. Balancing different objectives by weight functions is at best controversial. Fuzzy logic is a convenient vehicle for trading off different objectives. It allows the mapping of values of different criteria into linguistic values which characterize the level of satisfaction of the designer with the numerical value of objectives and operation over the interval [0,1] defined by the membership functions for each objective.

Three linguistic variables are defined to correspond to the three component objective functions: number-of-features f_1, number-of-incorrect predictions f_2, and average

classification error rate f_3. One linguistic value is defined for each component of the objective function. These linguistic values characterize the degree of satisfaction of the designer with the values of objectives $f_i(x)$, i = {1, 2, 3}. These degrees of satisfaction are described by the membership functions $\mu_i(x)$ on fuzzy sets of the linguistic values where $\mu(x)$ is the membership value for solution x in the fuzzy set. The membership functions for the minimum number of features, the minimum number of incorrect predictions, and the low classification error rate are easy to build. They are assumed to be non- increasing functions because the smaller the number of features $f_1(x)$, the number of incorrect predictions $f_2(x)$, and the classification error rate $f_3(x)$, the higher is the degree of satisfaction $\mu_1(x)$, $\mu_2(x)$, and $\mu_3(x)$ of the expert system (see Figure 1). The fuzzy subset of a good solution is defined by the following Fuzzy logic rule:
"**IF** a solution has *small number of features* AND *small number of incorrect predictions* AND *low classification error rate* **THEN** it is a good solution" According to the and/or like ordered-weighted-averaging Logic [17, 18], the above rule corresponds to the following:

$$\mu(x) = \gamma \times min(\mu_i(x)) + (1-\gamma) \times \frac{1}{3}\sum_{i=1}^{3}\mu_i(x). \quad (1)$$

where γ is a constant in the range [0,1]. The shape of the membershi function $\mu(x)$ is shown in Figure 1. Membership of data in a fuzzy set is defined using values in the range [0,1]. The membership values for the number of features F, the number of incorrect predictions P, and the classification error rate E are computed using equations 2, 3, and 4 respectively.

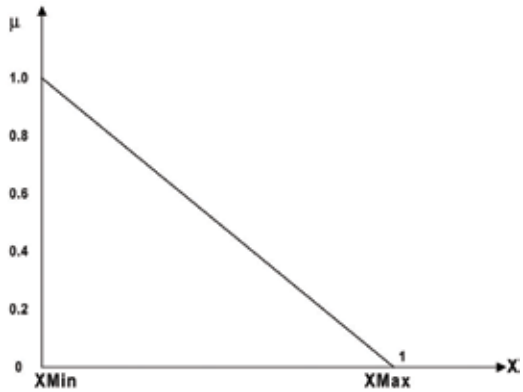


Fig. 1. Membership function for fuzzy subset X, where, in this application, X is the number of features F, the number of incorrect predictions P, or the classi¯cation error rate E.

$$\mu_1(x) = \begin{cases} 1 & \text{if } F \leq FMin \\ \frac{FMax-F}{FMax-FMin} & \text{if } FMin \leq F \leq FMax \\ 0 & \text{if } FMax \leq F \end{cases}$$

$$\mu_2(x) = \begin{cases} 1 & \text{if } P \leq PMin \\ \frac{PMax-P}{PMax-PMin} & \text{if } PMin \leq P \leq PMax \\ 0 & \text{if } PMax \leq P \end{cases}$$

$$\mu_3(x) = \begin{cases} 1 & \text{if } E \leq EMin \\ \frac{EMax-E}{EMax-EMin} & \text{if } EMin \leq E \leq EMax \\ 0 & \text{if } EMax \leq E \end{cases}$$

The maximum number of features (*FMax*) is the size of the feature vector and the minimum number of features (*FMin*) is 1 or *F*=2. The maximum number of incorrect predictions (*PMax*) and the maximum classification error rate (*EMax*) is determined by applying Naive Bayes Classifier for the initial solution. The minimum number of incorrect predictions (*PMin*) is 0 while the minimum classification error rate (*EMin*) is 0%.

3 Feature selection using Tabu Search

3.1 Initial solution

Feature selection vector is represented by a 0/1 bit string where 0 shows the feature is not included in the solution while 1 shows the feature is included. All features are included in the initial solution.

3.2 Neighborhood solutions

Neighbors are generated by randomly adding or deleting a feature from the feature vector of size *n*. For example, if 11001 is the current feature vector, then the possible neighbors with a candidate list size of 3 can be 10001, 11101, 01001. Among the neighbors, the one with the best cost (i.e. the solution which results in the minimum value of Equation 1) is selected and considered as a new current solution for the next iteration.

3.3 Tabu moves

A tabu list is maintained to avoid returning to previously visited solutions. With this approach, if a feature (move) is added or deleted at iteration *i*, then adding or deleting the same feature (move) for *T* iterations (tabu list size) is Tabu.

3.4 Aspiration criterion

Aspiration criterion is a mechanism used to override the tabu status of moves. It temporarily overrides the tabu status if the move is su±ciently good. In our approach, if a feature is added or deleted at iteration *i* and this move results in a best cost for all previous iterations, then this feature is allowed to add or delete even if it is in the tabu list.

3.5 Termination rule

The most commonly used stopping criteria in TS are

- after a fixed number of iterations.
- after some number of iterations without an improvement in the objective function value.
- when the objective reaches a pre-specified objective value.

In our algorithm, termination condition is a predefined number of iterations.

3.6 Intensification of the search

For intensification, the search is accentuated in the promising regions of the feasible domain. Intensification is based on some intermediate-term memory. Since, the solution space is

extremely large (initial feature vector $n > 100$), it is important to intensify the search in the promising regions by removing poor features from the search space. The following steps are used to intensify the search

- STEP1: Store M best solutions in intermediate memory for T_1 number of iterations.
- STEP2: Remove features that are not included in the best M solutions for N times.
- STEP3: Re-run the tabu search with the reduced set of features for another T_2 iterations.
- STEP4: Repeat steps 1-3 until the optimal or near-optimal solution is achieved.

where values of M and N can be determined empirically through experiments. As an example, assume that the following four best solutions as shown in Figure 2 are found by tabu search during T_1 iterations. Features f_1 and f_3 are always used while feature f_5 is never used for good solutions. For $N = 2$, the reduced feature set consists of only f_1, f_2, f_3, and f_6. Thus, tabu search will search for the near-optimal solutions in reduced search space avoid visiting non promising regions.

f1	f2	f3	f4	f5	f6	f7
1	0	1	0	0	1	0
1	0	1	1	0	1	0
1	1	1	0	0	0	1
1	1	1	0	0	1	0
Σ	4	2	4	1	0	3	1

Fig. 2. An example showing intensification steps for tabu search. Σ is the number of occurrences of each feature in the best solutions.

4. Experiments

We have performed a number of experiments and comparisons on several public data sets from the UCI [21] and non-public data sets from DynaVis Project [22] in order to demonstrate the performance of the classification system using Tabu Search. A short description of the used benchmarks is mentioned in Table 1. Classification results have been obtained by using N- Fold Cross Validation. In N-Fold CV, each dataset is divided into N blocks using $N-1$ blocks as a training set and the remaining block as a test set. Therefore, each block is used exactly once as a test set.

In addition, comparisons with several feature selection algorithms were also performed as mentioned below.

- Sequential Forward Search (SFS) [11]: SFS is the simplest greedy search algorithm. It starts with an empty feature subset and sequentially add features that results in the highest objective criteria. The main disadvantage of SFS is that it is unable to remove features that become irrelevant after the addition of other features.
- Sequential Forward Floating Selection (SFFS) [7]: SFFS improved the SF method by introducing backward steps after each forward step as long as the objective criteria increases.
- TS1: Tabu Search without Intensification [8, 27]
- TS2: Tabu Search with Intensification [25]

Data sets	Name	Samples	Features	Classes
UCI	Australian	690	14	2
	Breast Cancer	569	30	2
	German	1000	20	2
	Ionosphere	351	34	2
	Musk	476	166	2
	Sonar	208	60	2
DynaVis	CD Print 1	1534	74	2
	CD Print 2	1534	74	2
	CD Print 3	1534	74	2
	CD Print 4	1534	74	2
	Egg	4238	74	2
	Rotor	225	72	2

Table 1. Data sets Description. S = Samples, F = Features, C = Classes.

Table 2 shows a comparison of feature selection algorithms (SFS an SFFS) with TS-1 and TS-2 for UCI Datasets. Naive Bayes is used as a base classifier. From the table, it is clear that feature selection using TS has achieved higher accuracy when compared with SFS and SFFS in all data sets where $F > 50$. The most significant improvements are in terms of number of features. For German, Breast Cancer, and Ionosphere; only 9, 8 and 11 fea- tures are used respectively without comprising any accuracy when compared with SFS and SFFS. In Sonar, with only 18 features out of 60, the classification accuracy is 84.5% using TS-1 as compared to the accuracy of 80.8% and 82.2% using SFS and SFFS respectively. The number of features used by SFS and SFFS are 13 and 27 respectively for sonar. Similarly, in Musk, with only 37 features out of 166, the classification accuracy is 85.9% using TS-1 as compared to the accuracy of 83.2% and 84.0% using SFS and SFFS respectively. The number of features used by SFS and SFFS are 98 and 89 respectively for musk.

When compared tabu search with and without intensification; some improvement is achieved in terms of the number of features. For Australian; German and Breast Cancer; both TS-1 and TS-2 are identical. For Ionosphere, 11 features are used by TS-2 instead of 13 by TS-1 but with the same accuracy. In Sonar, with only 10 features out of 60, accuracy of 86.5% is achieved as compared to 18 features by TS-1 with accuracy 84.5%. In Musk, with only 36 features out of 166, accuracy of 86.8% is achieved as compared to 37 features by TS-1 with accuracy 85.9%.

Table 2 shows a comparison of feature selection algorithms (SFS and SFFS) with TS1 and TS2 for DynaVis Datasets. In all Data sets except rotor, some improvement in terms of accuracy and number of features are achieved using TS-2. In Rotor, with only 8 features out of 72, accuracy of 95.3% is achieved using TS-2 as compared to 92.4 (35 features) by SFS, 92.9 (8 features) by SFFS, and 95.1 (17 features) by TS-1.

Table 4 shows the computation cost for using various data sets and feature selection algorithms. The main computation cost is the evaluation of objective function or naive bayes classifier. Hence; number of evaluations are used as the main criterion to compare FS algorithms. As number of features increase; the number of evaluations are also increased. The number of iterations used for TS-1 are 500 for $F < 30$, 1000 for $F < 70$, and 2000 for $F > 70$. It should be noted that although it appears that TS-2/NB requires highest number of evaluations except in few cases when compared with SFFS but these evaluations are performed with fewer features. As an example; for Musk initially 26000 evaluations are used using 166 features, then 26000 but using 121 features, then 26000 using 104 features, and finally 26000 using only 92 features.

Data set	Method	Accuracy	Features Used	Standard Deviation
Australian	NB	80	14	3.40
	SFS/NB	87.7	7	4.00
	SFFS/NB	87.7	7	4.00
	TS-1/NB	87.7	7	4.00
	TS-2/NB	87.7	7	4.00
German	NB	73.6	20	5.32
	SFS/NB	75.9	10	4.53
	SFFS/NB	76.5	11	2.88
	TS-1/NB	76.5	9	3.63
	TS-2/NB	76.6	9	2.67
Breast Cancer	NB	93.3	30	2.09
	SFS/NB	97.4	5	2.48
	SFFS/NB	98.1	9	1.32
	TS-1/NB	98.1	8	1.32
	TS-2/NB	98.1	8	1.32
Ionosphere	NB	82.0	33	4.61
	SFS/NB	93.4	10	4.48
	SFFS/NB	93.7	12	4.22
	TS-1/NB	93.7	13	4.22
	TS-2/NB	93.7	11	4.22
Sonar	NB	68.3	60	11.39
	SFS/NB	80.8	13	7.00
	SFFS/NB	82.2	27	8.91
	TS-1/NB	84.5	18	8.60
	TS-2/NB	86.5	10	7.82
Musk	NB	73.3	166	4.08
	SFS/NB	83.2	98	4.85
	SFFS/NB	84.0	89	5.23
	TS-1/NB	85.9	37	4.30
	TS-2/NB	86.8	36	3.98

Table 2. Comparison of TS/NB with NB, SFS/NB, and SFFS/NB Classifier for UCI Datasets.

Two run time parameters i.e. the tabu list size and Number of neighbour- hood Solutions are determined using the following equation:

$$T=V^*=\text{ceil}(\sqrt{F}) \tag{5}$$

where T is the Tabu List Size, V^* is the number of neighbourhood solutions and F is the number of features.

5. Other advanced Tabu Search approaches for supervised classification

5.1 Simultaneous feature selection and feature weighting using hybrid tabu search/ *K*-nearest neighbor classifier

In [24], a hybrid tabu search/*K-NN* algorithm is proposed to perform both feature selection and feature weighting simultaneously with the objective of improving the classification

Data set	Method	Accuracy	Features Used	Standard Deviation
CD Print1	NB	87.2	74	2.63
	SFS/NB	92.3	8	1.91
	SFFS/NB	92.6	14	1.87
	TS-1/NB	92.5	21	1.69
	TS-2/NB	92.8	21	1.70
	TS-2/NB	92.1	8	2.00
CD Print2	NB	92.6	74	2.04
	SFS/NB	95.6	10	1.54
	SFFS/NB	95.7	14	1.32
	TS-1/NB	95.7	11	1.39
	TS-2/NB	95.8	13	1.40
	TS-2/NB	95.7	8	1.46
CD Print3	NB	89.0	74	2.43
	SFS/NB	92.7	14	1.98
	SFFS/NB	92.8	12	1.94
	TS-1/NB	93.1	15	1.86
	TS-2/NB	93.4	9	1.83
CD Print4	NB	90.4	74	1.00
	SFS/NB	93.2	9	1.20
	SFFS/NB	93.2	6	1.23
	TS-1/NB	93.6	12	1.28
	TS-2/NB	93.7	14	1.29
	TS-2/NB	93.2	6	1.25
Egg	NB	86.6	74	7.49
	SFS/NB	92.3	11	4.47
	SFFS/NB	92.4	14	4.60
	TS-1/NB	92.5	13	4.82
	TS-2/NB	92.5	13	4.82
	TS-2/NB	92.4	11	4.94
Rotor	NB	82.7	71	8.02
	SFS/NB	92.4	35	6.79
	SFFS/NB	92.9	8	7.46
	TS-1/NB	93.8	20	6.84
	TS-2/NB	95.1	17	5.85
	TS-2/NB	93.3	8	7.22

Table 3. Comparison of TS/NB with NB, SFS/NB, and SFFS/NB Classifier for DynaVis Datasets.

accuracy. This approach uses both a feature weight vector and a feature binary vector on the encoding solution of tabu search. The feature weight vector consists of real values while feature binary vector consisting of either 0 or 1. A *K*-NN classifier is used to evaluate each weight set evolved by TS. In addition to feature weight and binary vectors, the value of *K* used in *K*-NN classifier is also stored in the encoding solution of TS. Neighbors are calculated using an squared Euclidean distance defined as:

$$D(x,y) = \sum_{i=1}^{m} (x_i - y_i)^2 \tag{6}$$

where x and y are two input vectors and m is the number of features.

Dataset	NB	SFS/NB	SFFS/NB	TS-1/NB Iterations * $(\sqrt{F}+1)$	TS-2/NB TS-1/NB × {2,3,4}
Australian	1	106	326	2000	4000
German	1	211	929	2500	5000
Breast Cancer	1	466	2902	6000	12000
Ionosphere	1	562	12705	6000	12000
Sonar	1	1831	13637	8000	24000
Musk	1	13862	182399	26000	104000
CD Print 1	1	2776	73472	18000	54000
CD Print 2	1	2776	26472	18000	54000
CD Print 3	1	2776	23232	18000	54000
CD Print 4	1	2776	19902	18000	54000
Egg	1	2776	376593	18000	54000
Rotor	1	2629	53811	18000	54000

Table 4. Computation cost for various feature selection algorithms.

The classification accuracy obtained from TS/*K*-NN classifier is then compared and assessed with published results of several commonly-employed pattern classification algorithms. The results have indicated that simultaneous feature selection and weighting not only have the ability to find weights for *K*-NN classifier that result in higher classification accuracy but also have the ability to reduce the size of feature vector.

5.2 Round robin Tabu Search algorithm for multi-class problems

In [23], a novel round robin classification algorithm using a Tabu Search/Nearest Neighbor (TS/1NN) classifier to improve the classification accuracy of multi- class problems. Round robin classification is a technique which is suitable for use in multi-class problems. The technique consists of dividing the multi-class problem into an appropriate number of simpler binary classification problems [26]. Each binary classifier is implemented as TS/1NN classifier and the final outcome is computed using a simple voting technique. A key characteristic of this approach is that, in a binary class, the classifier tries to find features that distinguish only that class. Thus, different features are selected for each binary classifier, resulting in an overall increase in classification accuracy. In contrast, in a multi-class problem, the classifier tries to find those features that distinguish all classes at once. Results have indicated a significant increase in the classification accuracy for prostate cancer multi-class problem.

5.3 Feature selection for heterogeneous ensembles of nearest neighbour classifiers using hybrid Tabu Search

A new ensemble technique is proposed in [29, 28] to improve the performance of nearest neighbour (NN) classifier. This approach combines multiple NN classifiers, where each classifier uses a different distance function and potentially a different set of features (feature vector). These feature vectors are determined using a combination of Tabu Search (at the level of the ensemble) and simple local neighbourhood search (at the level of the individual classifiers). This ensemble classifier is evaluated using various benchmark data sets from UCI Machine Learning Repository. Results indicate a significant increase in the performance when compared with different well-known classifiers.

6. Conclusion

Feature selection (FS) algorithms are popular methods to reduce the dimensionality of the feature space and remove the redundant, irrelevant or noisy data and improves the classification accuracy. In this chapter; we have dis- cussed our recently proposed tabu search based algorithms for feature se- lection problems and have compared with other sequential feature selection algorithms. Tabu search is refined with respect to traditional approaches: A feature search intensification strategy is introduced. Search intensification is realized by incorporation of intermediate term memory in the search process. We have also discussed advanced tabu search approaches for supervised classification. Like many combinatorial optimization problems such as VLSI, Time Tabling, and Scheduling in which TS is quite useful, TS is also quite effective in data mining feature selection problems.

7. Acknowledgement

This work is supported by the European Commission (Project No. STRP016429, acronym DynaVis). This publication reflects only the author view.

8. References

A. K. Jain, R. P. W. Duin, and J. Mao. Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1), 4-37, 2000.

M. Kudo and J. Sklansky. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33, 25-41, 2000.

E. Amaldi, and V. Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. Theoretical Computer Science, 209, 237-260, 1998.

S. Davies, and S. Russell. NP-completeness of searches for smallest possible feature sets. *In Proceedings of the AAAI Fall Symposium on Relevance*, AAAI Press, pp. 37-39, 1994.

T. M. Cover, J. M. Van Campenhout. On the possible orderings in the measure- ment selection problem. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(9), 657-661, 1977.

A. K. Jain and D. Zongker. Feature selection: Evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine In- telligence*, 19(2), pp. 153-158, 1997.

P. Pudil, J. Novovicova, and J. Kittler. Floating search methods in feature selec- tion. *Pattern Recognition Letters*, 15, pp. 1119-1125, 1994.

H. Zhang and G. Sun. Feature selection using tabu search method. *Pattern Recognition*, 35, 701-711, 2002.

W. Siedlecki and J. Sklansy. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters*, 10(11), 335-347, 1989.

S. B. Serpico, and L. Bruzzone. A new search algorithm for feature selection in Hyperspectral Remote Sensing Images. *IEEE Transactions on Geoscience and Remote Sensing*, 39(7), 1360-1367, 2001.

A. W. Whitney. A direct method of nonparametric measurement selection. *IEEE Transactions on Computers*, 20(9), 1100-1103, 1971.

G. A. Clark et al. Multispectral image feature selection for land mine detection. *IEEE Transactions on Geoscience and Remote Sensing.* 38(1), pp. 304-311, 2000.

S. M. Sait and H. Youssef. General Iterative Algorithms for Combinatorial Optimization. IEEE Computer Society, 1999.

S. Yu, S. D. Backer, and P. Scheunders. Genetic feature selection combined with composite fuzzy nearest neighbor classifiers for hyperspectral satellite imagery. *Pattern Recognition Letters*. 23, pp. 183-190, 2002.

F. Glover. Tabu search I. *ORSA Journal on Computing*, 1(3), 190-206, 1989.

F. Glover (1990). Tabu search II. *ORSA Journal on Computing*, 2(1), 4-32, 1990.

H. J. Zimmerman. Fuzzy Set Theory and Its application. Kluwer Academic Publishers, 3rd Edition, 1996.

Salman A. Khan, Sadiq M. Sait, Habib Youssef. Topology Design of Switched Enterprise Networks Using Fuzzy Simulated Evolution Algorithm. *Engineering Applications of Artificial Intelligence*, Elsevier Science Ltd, pp. 327-340, 2002.

T. M. Cover, and P. E. Hart. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, 13(1), 21-27, 1967.

M. L. Raymer et al. Dimensionality Reduction using Genetic Algorithms. *IEEE Transactions on Evolutionary Computation*, *4(2)*, 164-171, 2000.

C. Blake, E. Keogh, and C. J. Merz. *UCI Repository of machine learning databases*. University of California, Irvine.

URL: www.dynavis.org

M. A. Tahir et al. *Novel Round-Robin Tabu Search Algorithm for Prostate Can- cer Classification and Diagnosis using Multispectral Imagery*. IEEE Transactions on Information Technology in Biomedicine, *10(4)*, 782-793, 2006.

M. A. Tahir, A. Bouridane, and F. Kurugollu. *Simultaneous feature selection and feature weighting using Hybrid Tabu Search/K-nearest neighbor classifier*. Pattern Recognition Letters, 28, 2007.

M. A. Tahir et al. A Novel Prostate Cancer Classification Technique using Intermediate Memory Tabu Search *Eurasip Journal on Applied Signal Processing, Special Issue: Advances in Intelligent Vision Systems: Methods and Applications*, 14, 2241-2249, 2005.

J. Furnkranz. Round Robin Classification. *Journal of Machine Learning Re- search*, 2, 721-747, 2002.

M. A. Tahir et al. Feature Selection using Tabu Search for Improving the Classification Rate of Prostate Needle Biopsies. *IEEE Proceedings of the International Conference on Pattern Recognition*, Cambridge, 2004.

M. A. Tahir and J. Smith. Feature Selection for Heterogeneous Ensembles of Nearest Neighbour Classifiers using Hybrid Tabu Search Advances in Metaheuristics for Hard Optimization, Springer Verlag, 2008.

M. A. Tahir and J. Smith. Improving Nearest Neighbor Classifier using Tabu Search and Ensemble Distance Metrics Proceedings of the sixth IEEE International Conference on Data Mining (ICDM), HongKong, 2006.

D. A. Bell and H. Wang A formalism for relevance and its application in Feature Subset Selection, Machine Learning, 41, 175-195, 2000.

R. Kohavi and G. John Wrappers for Feature Subset Selection, Artificial Intelligence, 97(1{2), 273-234, 1997.

G. H. John and R. Kohavi and K. Pfleger Irrelevant features and the subset selection problem, Proceedings of the Eleventh International Conference on Machine learning, New Brunswick,NJ, Morgan Kaufmann, 121-129, 1994.

A. Blum and P. Langley Selection of Relevant Features and Examples in Machine Learning, Artificial Intelligence, 97(1-2), 245-271, 1997.

11

Hybrid Tabu Search for Re-Entrant Permutation Flow-Shop Scheduling Problem

Jen-Shiang Chen[1], Jason Chao-Hsien Pan[2] and Chien-Kuang Wu[2]
[1]*Far East University,*
[2]*National Taiwan University of Science and Technology,*
Taiwan,
R.O.C

1. Introduction

The assumption of classical shop scheduling problems that each job visits each machine only once (Baker, 1974) is often violated in practice. A new type of manufacturing shop, the re-entrant shop has recently attracted attention. The basic characteristic of a re-entrant shop is that a job visits certain machines more than once. For example, in semiconductor manufacturing, consequently, each wafer re-visits the same machines for multiple processing steps (Vargas-Villamil & Rivera, 2001). The wafer traverses flow lines several times to produce the different layer on each circuit (Bispo & Tayur, 2001). A re-entrant flow-shop (RFS) refers to situations in which every job must be processed on machines in the order, M_1, M_2, …, M_m, M_1, M_2, …, M_m, …, and M_1, M_2, …, M_m. Every job can be decomposed into several layers each of which starts on M_1 and finishes on M_m. In the RFS case, if the job ordering is the same on any machine at each layer, then no passing is said to be allowed, since no job is allowed to pass any former job. The RFS scheduling problem in which no passing is allowed, is called a re-entrant permutation flow-shop (RPFS) problem.
The assumptions made for the RPFS scheduling problems are summarized here. Every job may visit certain machines more than once. Machine order is the same for each of the n jobs. Job order is the same for each of the m machines at each layer. The classical permutation flow-shop scheduling problem can be modified to suit the RPFS scheduling problem by relaxing the assumption that each job visits each machine no more than once. This study considers the RPFS scheduling problems with the objective of minimizing makespan of jobs.
Hwang & Sum (1998) addressed a two-machine flow-shop problem with re-entrant workflows and sequence dependent setup times, which have a special structure, to minimize makespan. Demirkol & Uzsoy (2000) proposed a decomposition method to minimize maximum lateness for a RFS with sequence-dependent setup times. Graves et al. (1983) modeled a wafer fabrication as a RFS, where the objective is to minimize average throughput time subject to meeting a given production rate. Drobouchevitch & Strusevich (1999) developed a heuristic algorithm for the two-machine re-entrant shop problem to minimize the makespan. Kubiak et al. (1996) considered a class of re-entrant shops in which jobs followed the route of M_1, M_2, M_1, M_3,…, M_1, M_m, M_1 with the objective of minimizing the mean flow time. They showed that the shortest-processing-time (SPT) rule was optimal

provided certain restrictive conditions held. Wang et al. (1997) proposed the scheduling of a chain-reentrant shop in which each job is first processed on a machine called the primary machine, then on other machines in a fixed sequence, and finally back to the primary machine for last operation. The objective of the problem is to minimize the makespan.

Tabu search (TS) is a meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality. The local procedure is a search that uses an operation called move to define the neighborhood of any given solution. One of the main components of TS is its use of adaptive memory, which creates a more flexible search behavior. Memory-based strategies are the hallmark of TS approaches (Glover & Languna, 1997). It has been shown to be a remarkably effective approach in a wide spectrum of problem areas from general integer and nonlinear programming to sequencing and production scheduling problems. Tabu search is a local search based optimization method that has been successfully used to solve many difficult combinatorial optimization problems, particularly in the scheduling area. These methods suggested by Glover (1989) can be sketched as follows: starting from an initial feasible solution, at each step we choose a move to a neighboring solution in such a way that we move stepwise towards a solution giving hopefully the minimum value of some objective function. Nowicki & Smutnicki (1996, 1998) developed effective TS methods for job-shop, flow-shop, and flow-shop with parallel machines problems to optimize the makespan criterion. These algorithms employ a classical insertion neighborhood, which is significantly reduced by a candidate list strategy for removing useless moves, in order to concentrate on "the most promising part" of the neighborhood.

As to the $n/m/J/C_{\max}$ problem which has been studied for a long time and is known to be NP-hard (Garey et al., 1976), the algorithm given by Adams et al. (1988), called shifting bottleneck uses the iterative solutions of a single bottleneck machine problem to build up and improve a schedule. Better solutions than the ones given by deterministic algorithms were found using simulated annealing but at the cost of longer computations. Tabu search was the first applied to job-shop by Taillard (1989), who proposed a sequential and a parallel algorithm. Dell'Amico & Trubian (1993) applied TS to the notoriously difficult job-shop scheduling problem.

For $n/m/F/C_{\max}$ problems, Palmer (1965) developed a quick method of obtaining a near optimum and Campbell et al. (1970) presented a heuristic algorithm as well. Widmer & Hertz (1989) used a simple insertion heuristic based on an analogy with the traveling salesman to the flow-shop problem to generate the starting order of the jobs and tried to improve this solution using TS techniques. In direct competition with the heuristic developed by Nawaz et al. (1983), TS method performed superiorly for 58% of the problems and matched the best solutions found for 92% of the problems.

Pan & Chen (2003) presented three extended mixed binary integer programming formulations and six extended effective heuristics for solving RPFS scheduling problems to minimize makespan. The TS method has been used to solve classical flow-shop problems and has performed well. This study considers RPFS scheduling, and applies hybrid tabu search (HTS) to minimize the makespan of jobs. The hybridization method is used to improve pure TS performance. The HTS is compared to the optimal solutions generated using the integer programming technique (Pan & Chen, 2003), and to the near optimal solutions generated by pure TS and other heuristics proposed by Pan & Chen (2003).

2. The optimization model

A classical (permutation) flow-shop problem assumes that all operations of each job visit every machine exactly once in the order of M_1, M_2, …, and M_n. Define this order of processing to be a level, then the routing requirement of a job in a RPFS problem can be decomposed into several levels. Hence, a classical permutation flow-shop is a special case of a RPFS with a single level and some of its formulations can be extended to solve the RPFS. To illustrate the concept of level decomposition, consider job i consisting of six operations to be processed on two machines, where (i, j, k) denotes that operation j of job i must be processed on M_k and thus its routing is (i, 1, 1)→ (i, 2, 2)→ (i, 3, 1)→ (i, 4, 2)→ (i, 5, 1)→ (i, 6, 2) and the corresponding processing time of each operation is orderly 8, 2, 7, 4, 5, and 1. The processing requirement of job i can be decomposed into three levels, where (i, 1, 1)→ (i, 2, 2) is the first level, (i, 3, 1)→ (i, 4, 2) is the second, and (i, 5, 1)→ (i, 6, 2) is the third. Let O^i_{lk} be the operation of job i on machine k at level l, p^i_{lk} be the processing time of the operation of job i on machine k at level l. Consequently, $O^i_{11} = (i, 1, 1)$, $O^i_{12} = (i, 2, 2)$, $O^i_{21} = (i, 3, 1)$, $O^i_{22} = (i, 4, 2)$, $O^i_{31} = (i, 5, 1)$, $O^i_{32} = (i, 6, 2)$, $p^i_{11} = 8$, $p^i_{12} = 2$, $p^i_{21} = 7$, $p^i_{22} = 4$, $p^i_{31} = 5$, and $p^i_{32} = 1$.

2.1 Notations

M = a very large positive number;
m = number of machines in the shop;
n = number of jobs for processing at time zero;
L = number of levels of job i;
p^i_{lk} = the processing time of the operation of job i on machine k at level l;
x_{ij} = 1, if job i is scheduled in the jth position at each level; 0, otherwise;
h_{klj} = the starting time of the operation scheduled at jth position of level l on machine k;
$C_{\max}$ = the maximum completion time or makespan;

2.2 Formulation

Pan & Chen (2003) were the first authors to present the integer programming model for solving the reentrant permutation flow-shop problem. The binary variable x_{ij} that the model uses is restricted by a single permutation of the numbers 1, 2, …, n that specifies the order in which jobs are processed on any machine at each level. The model is as follows.
Minimize

$$C_{\max} \tag{1}$$

Subject to

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, 2, ..., n \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad j = 1, 2, ..., n \tag{3}$$

$$h_{111} = 0 \tag{4}$$

$$h_{1,1,j+1} = h_{11j} + \sum_{i=1}^{n} x_{ij} p^i_{11} \quad j = 1, 2, \ldots, n-1 \tag{5}$$

$$h_{1,l,j+1} \geq h_{1lj} + \sum_{i=1}^{n} x_{ij} p_{l1}^{i} \quad l = 2, 3, \ldots, L; j = 1, 2, \ldots, n-1 \tag{6}$$

$$h_{1,l+1,1} \geq h_{1ln} + \sum_{i=1}^{n} x_{in} p_{l1}^{i} \quad l = 1, 2, \ldots, L-1 \tag{7}$$

$$h_{1,l+1,j} \geq h_{mlj} + \sum_{i=1}^{n} x_{ij} p_{lm}^{i} \quad l = 1, 2, \ldots, L-1; j = 1, 2, \ldots, n \tag{8}$$

$$h_{k,l,j+1} \geq h_{klj} + \sum_{i=1}^{n} x_{ij} p_{lk}^{i} \quad k = 2, 3, \ldots, m; l = 1, 2, \ldots, L; j = 1, 2, \ldots, n-1 \tag{9}$$

$$h_{k,l+1,1} \geq h_{kln} + \sum_{i=1}^{n} x_{in} p_{lk}^{i} \quad k = 2, 3, \ldots, m; l = 1, 2, \ldots, L-1 \tag{10}$$

$$h_{k+1,1,1} = h_{k11} + \sum_{i=1}^{n} x_{i1} p_{1k}^{i} \quad k = 1, 2, \ldots, m-1 \tag{11}$$

$$h_{k+1,l,j} \geq h_{klj} + \sum_{i=1}^{n} x_{ij} p_{lk}^{i} \quad k = 1, 2, \ldots, m-1; l = 1, 2, \ldots, L; j = 1, 2, \ldots, n;$$

$$(l, j) \notin \{(1, 1)\} \tag{12}$$

$$h_{k+1,l+1,1} \geq h_{kln} + \sum_{i=1}^{n} x_{in} p_{lk}^{i} \quad k = 1, 2, \ldots, m-1; l = 1, 2, \ldots, L-1 \tag{13}$$

$$C_{\max} = h_{mLn} + \sum_{i=1}^{n} x_{in} p_{Lm}^{i} \tag{14}$$

$$C_{\max} \geq 0, h_{klj} \geq 0 \quad k = 1, 2, \ldots, m; l = 1, 2, \ldots, L; j = 1, 2, \ldots, n;$$

$$x_{ij} = 0 \text{ or } 1 \quad i = 1, 2, ..., n; j = 1, 2, ..., n \tag{15}$$

Constraint (1) describes the objective function. Constraints (2) to (5) and (11) are essentially definitional, while constraints (6) to (10), (12) and (13) enforce the precedence relationships. Constraint (14) defines $C_{\max}$ to be the finish time of the last job processed on M_m at the last level. The non-negativity and binary restrictions on h_{klj} and x_{ij}, respectively, are specified in (15).

3. Hybrid tabu search

The HTS method differs from pure TS that it is not likely to trap in local optimum. The main idea of HTS is that when neighboring solutions are not able to update the current best solution for a period of time, a good problem-specific heuristic or dispatching rule is combined in pure TS to explore new solution region. With this feature, HTS is able to avoid falling into local optimum and move toward a better solution.

3.1 Initial solution

The classical (permutation) flow-shop problem has been proved to be NP-complete (Coffman, 1976; Rinnooy Kan, 1976). Hence, many heuristics have been proposed to provide a quick and good solution. Some of the well-known heuristics include the methods proposed by Campbell et al. (1970), Dennenbring (1977), Johnson (1954), Nawaz et al. (1983), and Palmer (1965). Pan & Chen (2003) made appropriate modifications to these six heuristics to solve the RPFS scheduling problems by taking the reentry property into account. The results showed that heuristic NEH (Nawaz et al., 1983) outperform the other algorithms in the set of problems with unknown optimal solutions. Hence, NEH is used to generate initial solution for RPFS problems.

3.2 Neighborhood search

Neighborhood search starts from current solution and seeks to find feasible, hopefully better, solutions in its neighborhood. If the neighboring solution is better than current one, this current solution is replaced by the neighboring solution until stopping rules satisfied. When dealing with RPFS problems, we have to focus on the jobs. The main reason is that once the processing sequence is determined, every machine follows the same order for all jobs. The problem will be simpler when we focus on jobs instead of operations. The neighborhood solutions are produced by interchanging the job order of the initial solution.

3.3 Choosing a move

First, the makespan for each neighborhood solution is calculated. Second, the solution that has the minimal makespan among others and outside tabu list or meets aspiration criterion is selected as a move.

3.4 Recording in tabu list

Nowicki & Smutnicki (1996) suggested recording the number of jobs exchanges of the move in tabu list. By doing so, whether two jobs had performed exchange or not can be held in the tabu list. A move $v = (x, y)$ is added to tabu list T in the following standard way. The tabu list T is shifted one position forward and put v in the last position in the list, that is, $T_j = T_{j+1}$, $j = 1, 2, \ldots, maxt - 1$, and $T_{maxt} = v$. In this study, the length of tabu list is set to seven and *first-in-first-out* (FIFO) rule is adopted; that is, when the tabu list is full, the new move replaces the earliest one entering tabu list and adds the maximal searching times by one.

3.5 Recording the best-so-far solution

If the solution after the move is better than the current best-so-far solution, replace the best-so-far solution and reset the non-improvement times to zero; otherwise, add non-improvement times by one.

3.6 A hybrid method

When a new best solution cannot be found for longer than a predetermined number of iterations, that is, count > threshold, the search switches to heuristic phase. Normal TS in this situation usually calls for intensification or diversification strategies to get out of a local optimum. Typical intensification or diversification strategies keep memory structures for storing rather a rather long history of recent search activities and use these structures to

guide future search directions (Hwang et al., 2002). The idea of Hwang et al. (2002) is cited to find the hybrid occasion of TS and heuristic. When the non-improvement times increase continuously, it means that best-so-far solution is not replaced by neighborhood solutions for a period of time, which is a signal that TS is likely to entrap in local optimum. In this situation, a hybrid method is introduced to explore new solution region. A threshold is set to twenty, which means that once the non-improvement times were cumulated to twenty, NEH is hybridized into TS to find a new solution. After that, non-improvement times are reset to zero and the searching process proceeds based on the new solution until stopping rules are satisfied. The overall procedure for the HTS algorithm is as follows.

Pseudo-code for the HTS algorithm

Find an initial solution x
Define tabu structure and set *Count* = 0
Repeat until stopping condition is met
Generate neighborhood sets of x: S_1, S_2, ..., S_k
Select the best non-tabu solution x' from $S_1 \cup S_2 \cup \ldots \cup S_k$
$x \leftarrow x'$
if x is better than the current best-so-far solution **then**
  Count = 0
else
  Count = *Count* + 1
end if
if *Count* > threshold **then**
  Update x by calling NEH heuristic and set *Count* = 0
end if

This hybrid method is illustrated in Fig. 1. Suppose the sequence of a schedule is (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12), two position C_1, C_2 are selected randomly from 1 to 12, representing the starting and end point of the substring (C_1, C_2). This substring is then treated as a sub-problem and solved by NEH heuristic. The new sequence then replaces the original substring (C_1, C_2). By doing this, a new solution is generated and serves as a new starting point of TS in order to get rid of local optimum.

3.7 Stopping rules

There are two stopping rules considered in this study and they are stated below.

(1) Non-improvement times:
This rule counts the number of non-improvement moves for TS. When the best-so-far solution cannot be replaced after one iteration, this counter adds by one.

(2) Max iteration:
This is the maximal iteration number that a TS takes. Once this number is reached, the TS is terminated.

4. An illustrative example

A small size problem of RPFS is given in this section. In the example, it is assumed that there are five jobs (n = 5), three machines (m = 5), and each job reenter twice (L = 2) in the shop. In a RPFS problem, the job sequence on each machine is the same and any sequence change on one machine will result in the sequence change in the rest machines.

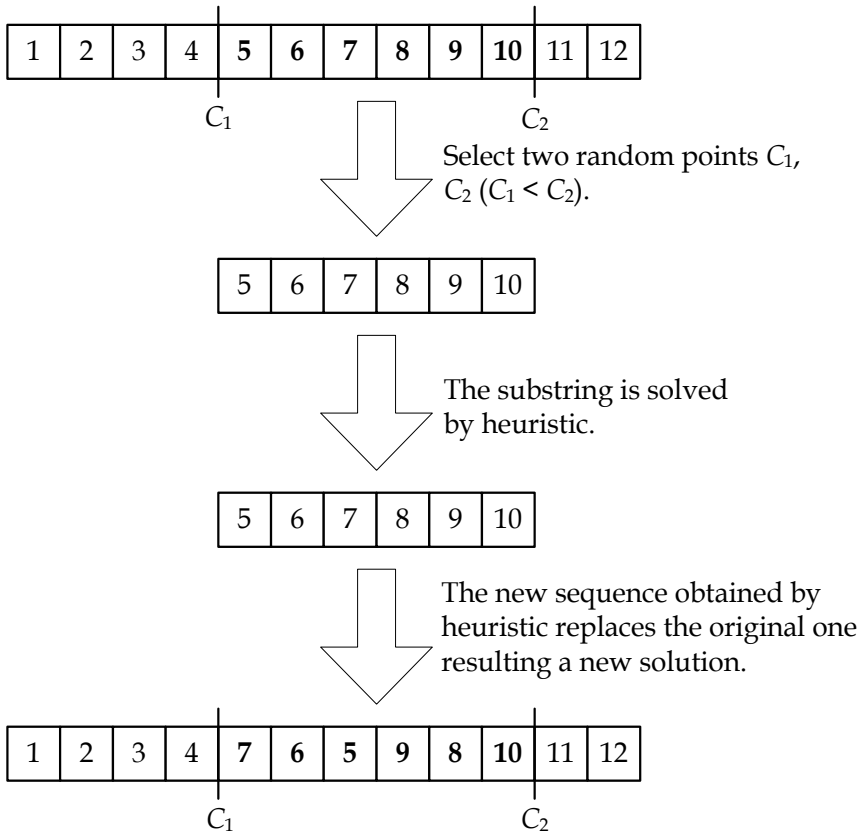


Fig. 1. A hybrid tabu search

4.1 Initial solution

In RPFS problems, NEH heuristic is used to generate an initial solution. For example, the schedule by NEH is (3, 5, 1, 4, 2) which represents the job sequence on each machine in the shop. If this sequence is changed, the processing order of jobs on each machine changes accordingly.

4.2 Neighborhood search

First, (3, 5, 1, 4, 2) is the starting point and the pair-wise exchange method is applied on it, as shown in Table 1. Next, the neighboring solution with the least makespan value and outside of the tabu list is selected as a move.

4.3 Record into tabu list

It is found that neighboring solution 2 has the minimal makespan among these neighborhood solutions (see Table 1), and this schedule is obtained by exchanging job 3 and job 1. Therefore, it is needed to check whether job 3 and job 1 are in tabu list. If they are not in tabu list, a move is made and iteration number is added by one; otherwise, the

neighboring solution with second least makespan is checked. If it is in tabu list, too, the third least one is checked. This procedure continues until none of the exchanged operations are in tabu list and a move can be made.

Number	Neighborhood solution					Makespan
1	5	3	1	4	2	340
2	1	5	3	4	2	335
3	4	5	1	3	2	347
4	2	5	1	4	3	337
5	3	1	5	4	2	348
6	3	4	1	5	2	355
7	3	2	1	4	5	340
8	3	5	4	1	2	348
9	3	5	2	4	1	358
10	3	5	1	2	4	345

Table 1. The sequence and makespan of neighborhood solutions

4.4 Record best-so-far solution

Compare the makespan (= 335) of neighboring solution 2 to that of the best-so-far solution (= 343). If it is better than the best-so-far solution, update the best-so-far solution and reset the non-improvement times to zero; otherwise, the best-so-far solution is kept and non-improvement times is added by one.

4.5 Hybrid method

The hybrid method for RPFS is described briefly. First, the new solution (3-5-1-4-2) is use to search better neighborhood solutions until stopping rule are satisfied. In the following, several iterations are omitted to describe the hybrid method directly. In RPFS example, the threshold value is also set to 3. After several iterations, the neighboring solution (3-2-1-5-4) is generated. Then, two points are selected randomly, say $C_1 = 2$, $C_2 = 4$ to define an interval (i.e., a substring). This substring is then treated as a RPFS subproblem and rescheduled by NEH heuristic. The new sequence then replaces the original substring to form a new solution, as shown in Fig. 2. Finally, we base on this new solution to search new neighborhood solutions and find a neighboring solution with makespan of 327 is better than the best-so-far solution (= 335). These neighborhood solutions based on above searching procedures repeats until the stopping rules are satisfied.

5. Computational results

The experimental environment and the meaning of each parameter are described as follows. n is the number of jobs, m is number of machines, and L is number of layers. The problem n×m×L is a RPFS problem with n jobs, m machines, L layers. The test problems are classified

into 3 categories: small problems, medium problems, and large problems. Types of small problems include 3×3×3, 4×4×4, 5×4×3, 5×5×4, 6×8×5, 7×8×4, 8×8×4, 9×7×4, 9×9×3, and 10×6×3. Types of medium problems include 11×17×5, 12×20×6, 13×19×7, 14×18×9, 15×17×6, 16×16×7, 17×15×8, 18×16×6, 19×12×10, and 20×15×8. Types of large problems include 25×25×10, 30×30×7, 40×40×6, 50×50×5, 60×60×3. The processing time of each operation for each type of problem is a random number generated from [1, 100] since the processing times of most library benchmark problems are generated in this range (Beasly, 1990).

In order to demonstrate the performance of HTS, it is compared to optimal solution obtained by integer programming (IP) for small problems. The IP model is proposed by Pan and Chen (2003) for solving RPFS scheduling problems. For medium and large problems, HTS is compared to its initial solution or to the pure TS solution. In this study, IP model is solved by ILOG CPLEX software. The programs for heuristics are coded in Visual C++ language and implemented on PC with Pentium IV 1.6 GHz.

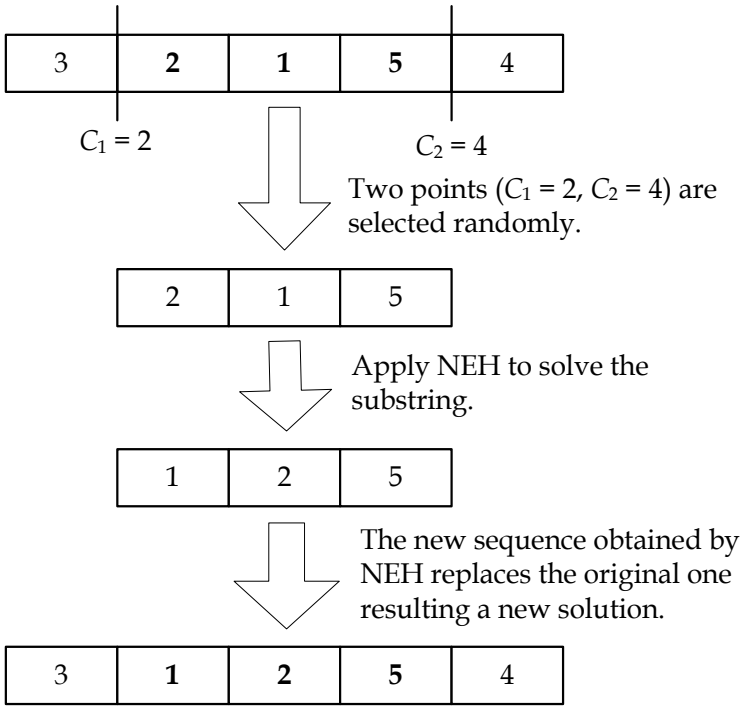


Fig. 2. The hybrid method of RPFS example

5.1 Small problems

In the experiment, ten instances are generated for each problem type and the average makespan is analyzed. For each problem type, the average makespan of HTS is compared to that of optimal makespan. The difference of these two average values is a measure of the efficiency of HTS. Similarly, HTS is also compared with its initial solution to obtain the improvement rate. The encoding scheme is based on jobs rather than on operations. The maximal number of iterations is set to 3,000 and non-improvement times are set to 1,500.

The results for the type of small problem are shown in Table 2. The solutions of HTS are also close to optimum (0.004% above optimum on the average). These results show that HTS is efficient and has good solution quality of less than 1% above optimal. Then the percentage error of HTS is defined as:

$$\text{Percentage error} = \frac{C_{\max}(\text{HTS}) - C_{\max}(\text{IP})}{C_{\max}(\text{IP})} \times 100\%$$

Where C_{max}(HTS) and C_{max}(IP) are the makepan obtained by HTS and IP, respectively.

5.2 The medium problems

For medium problems, the solutions obtained by HTS are compared to initial solutions and those obtained by pure TS, where the initial solutions are generated by heuristic NEH (Pan & Chen, 2003). The maximal iteration number is 2000 and non-improvement time is 1000. The comparison results of all medium problems are displayed in Table 3. Table 3 shows that the quality of solutions generated by HTS is 2.83% better than its initial solution obtained by NEH. Additionally, the performance of HTS is 0.57% better than pure TS.

5.3 Large problems

Large problems are tested with the same basis as those of medium problems and the types of large problems tested are shown in Table 4. The maximal iteration number is 1200 and non-improvement time is 600. The performance difference between HTS and NEH, HTS and pure TS is reported, respectively. It is shown that the solution quality of HTS is 2.53% better than its initial solutions generated by NEH. For comparison of the performance between HTS and pure TS, the efficiency of HTS is 0.81% better than that of the pure TS. It is noted that improvement rate increases as the number of jobs increases.

Types	IP	HTS	
	Time (s)	Time (s)	Avg. percentage error (%)
3×3×3	0.03	0	0
4×4×4	0.11	0.007	0.041
5×4×3	0.18	0.42	0
5×5×4	0.32	0.41	0
6×8×5	2.33	0.60	0
7×8×4	6.93	0.76	0
8×8×4	32.83	2.15	0
9×7×4	71.93	2.56	0
9×9×3	90.63	2.41	0
10×6×3	20.52	5.43	0

Table 2. Comparison of all small problems solved by IP and HTS

Types	CPU time(s)			Comparison	
	HTS	TS	NEH	The improvement rate of HTS over TS (%)	The improvement rate of HTS over NEH (%)
11×17×5	3.05	3.18	0.01	0.62	2.29
12×20×6	5.63	5.03	0.01	0.61	2.12
13×19×7	8.46	8.35	0.01	0.48	2.13
14×18×9	15.89	12.37	0.02	0.46	2.11
15×17×6	12.04	8.64	0.02	0.81	3.07
16×16×7	16.53	13.34	0.02	0.47	2.98
17×15×8	19.86	19.97	0.02	0.37	3.13
18×16×6	23.28	19.78	0.02	0.56	3.74
19×12×10	28.59	25.98	0.02	0.60	3.12
20×15×8	38.39	37.00	0.02	0.71	3.64

Table 3. The improvement results of all medium problems

Types	CPU time(s)			Comparison	
	HTS	TS	NEH	The improvement rate of HTS over TS (%)	The improvement rate of HTS over NEH (%)
25×25×10	155.22	132.67	0.02	0.78	2.53
30×30×7	242.60	179.28	0.21	0.79	2.63
40×40×6	612.24	481.94	0.31	0.81	2.43
50×50×5	1006.44	872.64	0.76	0.83	2.46
60×60×3	1195.09	1104.5	0.85	0.83	2.61

Table 4. The improvement results of all large problems

6. Conclusions and suggestions for future study

This study applies HTS to solve RPFS scheduling problems with objective to minimize makespan. In pure TS, if the solution cannot escape from local optimum, the improvement rate can hardly be increased even a great amount of computational time is spent. The proposed HTS is used to improve the efficiency of TS. The heuristic method is hybridized into pure TS to find better solution regions.

In RPFS, job-based encoding is adopted to deal with different types of problems. The results show that HTS obtains favorable solutions within reasonable time. For small problems, the percentage of HTS finding optimal solutions is near 100%. For medium problems,

comparisons are made between HTS and the initial solutions obtained by NEH. It is found that HTS improves the initial solution favorably. For large problems, HTS is superior to heuristic NEH. For the medium and large problems, HTS is compared to pure TS method. The results show that HTS is superior to pure TS. Moreover, it is found that the improvement rate of HTS over TS increases with the increase of problem size. Hence, it is clear that the incorporation of appropriate heuristic with pure TS is indeed effective.

Some future study suggestions are given as follows:

(1) A static tabu list is used in this study. A dynamic tabu list may be used in future study to investigate whether the solution quality can be improved.

(2) A thorough study of the effect of maximal iteration number and non-improvement times on solution quality may be carried out in future studies.

(3) Other exchanging method to obtain neighborhood solutions can be investigated and the techniques of experimental design may be applied to find out the best way of neighborhood search.

7. References

Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science*, Vol. 34, No. 3, 391-401, ISSN: 0025-1909.

Baker, K. R., (1974). *Introduction to sequencing and scheduling*, John Wiley & Sons, ISBN: 0-471-04555-1, New York.

Beasly, J. E. (1990). OR-library: distribution test problems by electronic mail. *Journal of the Operational Research Society*, Vol. 41, No. 11, 1069-1072, ISSN: 0160-5682.

Bispo, C. F. & Tayur, S. (2001). Managing simple re-entrant flow lines: theoretical foundation and experimental results. *IIE Transactions*, Vol. 33, No. 8, 609-623, ISSN: 0740-817X.

Bowman, E. H. (1959). The scheduling-sequence problem. *Operations Research*, Vol. 7, 621-624, ISSN: 0030-364X.

Campbell, H. G., Dudek, R. A., & Smith, M. L. (1970).. *Management Science*, Vol. 16, No. 10, B630-B637, ISSN: 0025-1909.

Coffman, E. G. (1976). Computer and Job Shop Scheduling. Wiley, ISBN: 0471163198, New York.

Dell'Amico, M., & Trubian, M. (1993). Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research*, Vol. 41, No. 3, 231-252, ISSN: 0254-5330.

Dannenbring, D. G. (1977). An evaluation of flow shop sequencing heuristics. *Management Science*, Vol. 23, No. 11, 1174-1182, ISSN: 0025-1909.

Demirkol, E. & Uzsoy, R. (2000). Decomposition methods for re-entrant flow shops with sequence-dependent setup times. *Journal of Scheduling*, Vol. 3, No. 3, 155-177, ISSN: 1094-6136.

Drobouchevitch, I. G., & Strusevich, V. A. (1999). A heuristic algorithm for two-machine re-entrant shop scheduling. *Annals of Operations Research*, Vol. 86, 417-439, ISSN: 0254-5330.

Garey, M. R.; Johnson, D. S. & Sethi, R. (1976). The complexity of flow-shop and job-shop scheduling. *Mathematics of Operations Research*, Vol. 1, No. 2, 117-129, ISSN: 0364-765X.

Glover, F. (1989). Tabu search– Part I. *ORSA Journal on Computing*, Vol. 1, 190-206, ISSN: 0899-1499.

Glover, F., & Languna, M. (1993). Tabu Search– Modern heuristic techniques for combinatorial problems. Colin R. Reeves (Ed.), 70-150, Blackwell Scientific Publications, ISBN: 0470220791, Oxford.

Glover, F., & Languna, M. (1997). Tabu Search. Colin R. Reeves (Ed.), Blackwell Scientific Publications, ISBN: 079239965X, Oxford.

Graves, S. C.; Meal, H. C.; Stefek, D. & Zeghmi, A. H. (1983). Scheduling of re-entrant flow shops. *Journal of Operations Management*, Vol. 3, No. 4, 197-207, ISSN: 0272-6963.

Hwang, H. and Sun, J. U. (1998). Production sequencing problem with re-entrant work flows and sequence dependent setup times. *International Journal of Production Research,* Vol. 36, No. 9, 2435-2450, ISSN: 0020-7543.

Hwang, J., Kang, C. S., Ryu, K. R., Han, Y., & Choi, H. R. (2002). A hybrid of tabu search and integer programming for subway crew paring optimization. Proceedings of the Sixth IASTED International Conference on Artificial Intelligence and Soft Computing (ASC-2002), 72-77, Banff, Canada.

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with set up times included, *Naval Research Logistics Quarterly*, Vol. 1, No. 1, 61-68.

Kubiak, W.; Lou, S. X. C. & Wang, Y. (1996). Mean flow time minimization in re-entrant job-shops with a hub. *Operations Research*, Vol. 44, No. 5, 764-776, ISSN: 0030-364X.

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine n-job flow-shop sequencing problem. *OMEGA*, Vol. 11, No. 1, 91-95, ISSN: 0305-0483.

Nowicki, E., & Smutnicki, C. (1996). A fast taboo search algorithm for the job shop problem. *Management Science*, Vol. 42, No. 6, 797-813, ISSN: 0025-1909.

Nowicki, E., & Smutnicki, C. (1996). A fast taboo search algorithm for the permutation flow-shop problem. *European Journal of Operational Research*, Vol. 91, No. 1, 160-175, ISSN: 0377-2217.

Nowicki, E., & Smutnicki, C. (1998). Flow shop with parallel machines: A tabu search approach. *European Journal of Operational Research*, Vol. 106, No. 2-3, 226-253, ISSN: 0377-2217.

Palmer, D. S. (1965). Sequencing jobs through a multi-stage process in the minimum total time– A quick method of obtaining a near optimum. *Operational Research Quarterly*, Vol. 16, No. 1, 101-107, ISSN: 0030-3623.

Pan, J. C. H. & Chen, J. S. (2003). Minimizing makespan in re-entrant permutation flow-shops. *Journal of the Operational Research Society*, Vol. 54, No. 6, 642-653, ISSN: 0160-5682.

Rinnooy Kan, A. H. G. (1976). *Machine scheduling problems: classification, complexity and computations*, Martinus Nijhoff, ISBN: 90.247.1848.1, The Hague, Holland.

Taillard, E. (1989). Parallel taboo search technique for the job shop scheduling problem. Internal Report ORWP89/11, Department de Mathematiques, Ecole Polytechnique Federale de Lausanne, Lausanne.

Vargas-Villamil, F. D. & Rivera, D. E. (2001). A model predictive control approach for real-time optimization of re-entrant manufacturing lines. *Computers in Industry*, Vol. 45, No. 1, 45-57, ISSN: 0166-3615.

Wang, M. Y.; Sethi, S. P. & Van De Velde, S. L. (1997). Minimizing makespan in a class of re-entrant shops. *Operations Research*, Vol. 45, No. 5, 702-712, ISSN: 0030-364X.

Widmer, M., & Hertz, A. (1989). A new heuristic method for the flow shop sequencing problem. *European Journal of Operational Research*, Vol. 41, 186-193, ISSN: 0377-2217.

12

On the Design of Large-scale Cellular Mobile Networks Using Tabu Search

Alejandro Quintero and Samuel Pierre
Mobile Computing and Networking Research Laboratory (LARIM)
Department of Computer Engineering, École Polytechnique de Montréal
Canada

1. Introduction

The design of UMTS networks involves optimizing a number of network configuration parameters in order to meet various service and performance requirements. Universal Mobile Telecommunications Service (UMTS) can be viewed as an evolution of the Global System for Mobile communication (GSM) that supports 3G services [18][22]. Generally, a UMTS network is divided into an access network and a core network (Figure 1). The former is dependent upon access technology, while the latter can theoretically handle different access networks. The access network is known as the UMTS Terrestrial Radio Access Network (UTRAN) and comprises two types of nodes: the Radio Network Controller (RNC) and the Node B, which is a base station (BS) [18][22]. It controls the radio resources within the network and can interface with one or more stations (Node Bs).

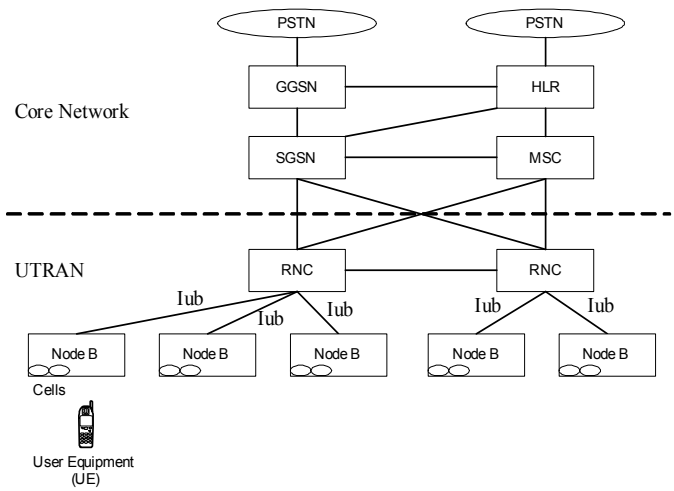


Figure 1. UMTS network architecture

The air interface used between the User Equipment (UE) and the UTRAN is Wideband Code-Division Multiple Access (WCDMA.) The UTRAN communicates with the core network over the Iu interface, which comprises two components: the Iu-CS interface,

supporting Circuit-Switched (CS) services and the Iu-PS interface for Packet-Switched (PS) services [18][22].

In the core network, Mobile Switching Centers (MSCs) are responsible for the circuit-switched location management, while Serving GPRS Support Nodes (SGSNs) assume the packet-switched location management. Both domains are linked through certain interfaces, but the information is kept in separate network entities: the circuit-switched location information remains in the MSC, while the packet-switched location information stays in the SGSN. The Home Location Register (HLR), a common location information database for both domains, contains the network's subscriber profiles. The Gateway GPRS Support Node (GGSN) links a UMTS network and a Public Switched Telephone Network (PSTN).

A Personal Communication Service (PCS) network, such as a UMTS, is a wireless communication network that integrates various services such as voice, video and electronic mail, which are accessible from a single mobile terminal and for which subscribers obtain a single invoice. These various services are offered in a cover zone area that is divided into Node Bs, which manage all the communications within the cell. In the cover zone, Node Bs are connected to special units called Radio Network Controllers (RNCs).

When a user's communication switches from one Node B to another, the new Node B becomes responsible for relaying this communication through the allocation of a new radio channel for the user. Supporting the transfer of the communication from one Node B to another is called a handover. This mechanism, which primarily involves the RNCs, occurs when the level of signal received by the user reaches a certain threshold. There are two types of handovers. In the case of Figure 2 for example, a movement where a user moves from Node B_i to Node B_j is referred to as a soft handover because these two nodes are connected to the same RNC. The RNC which supervises the two nodes remains the same and the cost is low in terms of system resources. On the other hand, a case where the user moves from Node B_i to Node B_k is considered a complex handover. The cost associated to this concept is superior as both RNC 1 and 2 remain active during the handover procedure and the database that contains subscribers' information requires an update.

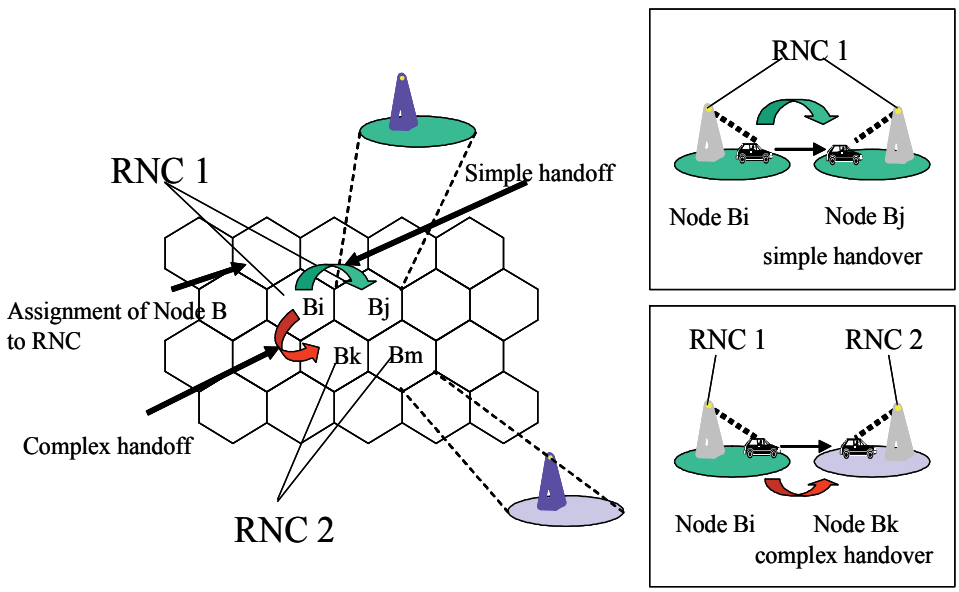


Figure 2. Geographic division in a cellular network

The total operating cost of a cellular network includes three components: the cost of the links between the Node Bs and the RNC to which they are joined, the monthly amortization cost of installed RNCs and the cost generated by the handovers between Node Bs. Therefore, intuitively, it seems more discriminating to join Nodes B_i and B_k to the same RNC, if their handover frequency is high. The problem of assigning Node Bs to RNCs consists essentially of finding the configuration that minimizes the total network operating costs. This problem can be solved by way of an exhaustive search method that would entail a combinatorial explosion and, therefore, an exponential augmentation of execution times [1][14].

This paper proposes a tabu search method to efficiently solve the problem of assigning Node Bs to RNCs in cellular mobile networks. Section 2 provides background and related work. Section 3 describes tabu search method. Section 4 presents some adaptation and implementation details. Finally, Section 5 elaborates on experimental results and compares them to other methods which are well documented in the literature.

2. Formulation of the problem and related work

The assignment problem consists of determining a Node Bs assignment pattern which minimizes a certain cost function, while respecting particular constraints, especially those related to limited RNC capacity [26][27].

Let n represent the number of Node Bs to be assigned to m RNCs. The location of Node Bs and RNCs is fixed and known.

Let H_{ij} depict the cost per time unit for a simple handover between Node Bi and Node Bj, involving only one RNC, and H'_{ij} the cost per unit of time for a complex handover between Node Bi and node Bj *(i, j = 1, ..., n with* $i \neq j$*)*, involving two different RNCs. H_{ij} and H'_{ij} are proportional to the handover frequency between Node Bi and Node Bj.

Let c_{ik} denote the amortization cost associated to the link between Node Bi and RNC k *(*$i = 1, \ldots, n$*;* $k=1, \ldots, m$*)*.

Let INS_k express the monthly amortization cost for each installed RNC k *(*$k=1, \ldots, m$*)*.

Let x_{ik} illustrate a binary variable, equal to 1 if Bs i is related to RNC k; otherwise x_{ik} is equal to 0.

The assignment of Bs to RNCs is subject to a number of constraints. In fact, each Node B must be assigned to a single RNC, which can be expressed as follows:

$$\sum_{k=1}^{m} x_{ik} = 1 \quad \text{for} \quad \text{i} = 1, \ldots, \text{n}. \tag{1}$$

Let z_{ijk} and y_{ij} be defined as:

$$z_{ijk} = x_{ik} x_{jk} \quad \text{for} \quad i, j = 1, \ldots, n \quad \text{and} \quad k = 1, \ldots, m, \quad \text{with} \quad i \neq j.$$

$$y_{ij} = \sum_{k=1}^{m} z_{ijk} \quad \text{for } i, j = 1, \ldots, n, \text{ and } i \neq j.$$

z_{ijk} is equal to 1 if Bi and Bj, with $i \neq j$, are both connected to the same RNC k; otherwise z_{ijk} is equal to 0. Thus, y_{ij} takes the value of 1 if Bi and Bj are both connected to the same RNC and the value 0 if Bi and Bj are connected to different RNCs.

The cost per unit of time *f* of the assignment is expressed as follows:
Minimize:

$$f = \sum_{i=1}^{n}\sum_{k=1}^{m} c_{ik}x_{ik} + \sum_{k=1}^{m} INS_k + \sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} H'_{ij}(1-y_{ij}) + \sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} H_{ij}y_{ij} \quad (2)$$

The first term of the equation represents the link cost. The second term takes into account the amortization cost of the installed RNC. The third term deals with the complex handover cost and the last, the cost of simple handovers. We must keep in mind that the cost function is quadratic in x_{ik}, as y_{ij} is a quadratic function of x_{ik}. It also bears mentioning that an eventual weighting could be taken into account, specifically in the link and handover cost definitions.
The capacity (calls per unit of time) of an RNC *k* is denoted M_k. If λ_i depicts the number of calls per unit of time directed to B*i*, the limited capacity of RNCs imposes the following constraint:

$$\sum_{i=1}^{n} \lambda_i x_{ik} \leq M_k \quad \text{for} \quad k = 1, ..., m \quad (3)$$

according to which the total load of all Node Bs that are assigned to the RNC *k*, is less than the capacity M_k of the RNC. Finally, the problem constraints are completed as follows:

$$x_{ik} = 0 \text{ or } 1 \text{ for } i=1, \ldots, n \text{ and } k = 1, \ldots, m. \quad (4)$$

$$z_{ijk} = x_{ij}x_{ik} \text{ and } i, j = 1, \ldots, n \text{ and } k = 1, \ldots, m. \quad (5)$$

$$y_{ij} = \sum_{k=1}^{m} z_{ijk} \quad \text{for} \quad i, j = 1, \ldots, n \quad i \neq j \quad (6)$$

Constraints (1), (3) and (4) pertain to transport problems. In fact, each Node B*i* could be likened to a factory that produces a call volume λ_i. The RNCs are then considered as warehouses of M_k capacity where Node Bs production could be stored.
The assignment problem consists of minimizing (2) under (1), and (3) to (6). When formulated that way, the problem cannot be solved with a standard method such as linear programming as Constraint (5) is not linear.
The total cost comprises three components, namely the handover cost between two adjacent Node Bs, installation cost of the RNC, and the cost of the link between Node Bs and RNCs. The design is to be optimized subject to the constraint that the call volume of each RNC must not exceed its call handling capacity. This kind of problem is an NP-hard problem [15], so enumerative searches are practically inappropriate for moderate- and large-sized cellular mobile networks [2][4][15][16]. Since they examine the entire search space in an exhaustive manner to find the optimal solution, they are only efficient for spaces which are relatively small, corresponding to small-sized instances of the problem. For example, for a network composed of *m* RNC and *n* Node Bs, m^n solutions must be examined.
Optimization problems arising during the various phases of a network life cycle differ, not only in their objectives, but also in the set of design parameters and the level of detail with which they deal. Mean site-to-site distances, site locations, sectorization, antenna types and average antenna heights are usually addressed in the dimensioning phase [2][25][28].

In [21], an engineering cost model is being proposed to estimate the cost for providing personal communications services in a new residential development. It estimates the expenses of building and operating a new Personal Communication Service (PCS) using existing infrastructures such as the telephone, cable television and cellular networks. In [8], the economic aspects of configuring cellular networks are presented. Major components of costs and revenues as well as the major stakeholders are identified and a model is developed to determine the system configuration (e.g., cell sizes, number of channels, link costs, etc.). For example, in a large cellular network, it is impossible for a Node B located in eastern United States to be assigned to an RNC located in western United States. In this case, the variable link cost is ∞. The geographical relationships between nodes and RNCs are considered in the link cost value, so that the Node Bs are generally assigned to neighboring RNCs and not to remote RNCs. In [9], various methods are suggested to estimate the handover rate in mobile networks and the economic impacts of mobility on system configuration decisions (e.g., annual maintenance and operations, channel costs, etc.) are addressed. The cost model proposed in this paper is based on [8][9][21].
Wu & Pierre [26][27] propose a strategy designed to solve this problem. The proposed hybrid search strategy is composed of three phases: a constraint satisfaction method with an embedded problem-specific goal to guide the search for an acceptable initial solution, an optimization phase using local search algorithms, such as tabu search [10] and a post optimization phase to improve the solutions generated by the second phase through a constraint optimization procedure. Merchant & Sengupta [15][16] studied this assignment problem. Their algorithm starts from an initial solution, which is improved through a series of greedy moves, while avoiding being stranded in a local minimum. The moves used to escape a local minimum explore only a very limited set of options. Such moves rely on the initial solution and do not necessarily generate an acceptable final solution. Beaubrun et al. [3] use simulated annealing to solve this problem in 2G. *Simulated Annealing* (SA), a stochastic algorithm, introduced by Metropolis et al. [17], is used to estimate the solution of very large combinatorial optimization problems. It represents an effective solution to certain combinatorial optimization problems. Other heuristic approaches have been developed for this kind of problem in 2G networks [6][7][19][20][23][25].

3. Tabu search, simulated annealing and genetic algorithm approaches

Different approaches can be used for assigning of Node Bs to RNC. One can mention heuristic approaches like tabu search, genetic algorithms and simulated annealing.

3.1 Tabu search approach

A *tabu search* method is an adaptive technique used in combinatorial optimization to solve NP problems [7][10]. Nevertheless, it is particularly the local search neighborhood and the way the tabu list is built and exploited that are subject to many variations, which accords tabu its meta-heuristic nature. The tabu list is not always a list of solutions: it can be a list of forbidden moves/perturbations [7]. However, accepting solutions that are not necessary the best introduce a cycle risk, i.e., a return to the solutions that have already been considered, hence the idea of keeping a *tabu list T* of solutions that have already been considered. Thus, during the generation of the set *V* of neighbour candidates, the solutions in the tabu list are not considered.

On one side, keeping a tabu list allows for avoiding the cycle risk; on the other side, the storage of the solutions that have already been found could require large amount of memory. Moreover, it proves useful to be able to return to a solution that has already been considered, in order to continue the search in another direction. A compromise is reached by keeping a tabu list, which avoids cycles with a length that is less or equal to *k*. However, a cycle with a length higher than *k* could always occur.

To minimize the objective function, the tabu search method starts from an initial solution and attempts to reach a global optimum by applying moves that allow the passage from a solution s_i to another solution s_{i+1} chosen in the neighborhood $N(s_i)$ of s_i (the $N(s_i)$ of a solution s_i is defined as the set of solutions that are accessible by applying one move to s_i). At each iteration *i*, the neighborhood where solutions are selected is redefined as the neighborhood of the current solution s_i. The tabu conditions change and the admissible solutions are no longer the same.

The exploration of the search domain could be represented by a graph $G=(X,A)$ where *X* refers to the set of solutions and *A* the set of arcs $(x, m(x))$, where $m(x)$ is the solution obtained by applying the move *m* to *x*. Then, a given move will be equivalent to a set of arcs $\{(x, m(x)), x \in X\}$. The graph *G* is symmetrical because for each arc $(x, m(x))$, there exists an arc $(m(x), x)$ obtained by applying the reverse move m^{-1} to $m(x)$. The tabu search method starts from an initial solution x_0, node of the graph *G*, and will search in *G* a path $x_0, x_1, \ldots, x_k$, where $x_i = m(x_{i-1})$ with $i=1,\ldots,k$. Each intermediary solution of the path is obtained by applying a move to the precedent solution. The arcs (x_i, x_{i+1}) of the path are chosen by solving the optimization problem:

$$f(x_{i+1}) = \min f(x_i) \text{ with } x_i \in V - T \tag{7}$$

where *V* in this case is the whole neighborhood $N(x_i)$ of x_i.

If the set *V-T* of solutions to be explored is too large, the cost induced by the algorithm could become prohibitive. Conversely, a too small set has the adverse effect on the quality of solutions found. The tabu search method could be distinguished from the general descending method, mainly by the use of a shorter-term memory structure, which allows for accepting less good solutions in order to exit local optima, while avoiding cycles. For more details on the tabu method, one can refer to [7].

To solve the assignment problem with a tabu search method, a search domain free from capacity constraints on the RNC was selected, while respecting the constraints of unique assignment of Node Bs to RNC. Two values are associated with each solution: the first one indicates the intrinsic cost of the solution, calculated from the objective function (equation 2) and the second expresses the solution evaluation, taking into account the costs and penalties for not respecting the capacity constraints. At each step, the solution associated with the best evaluation is chosen. Once an initial solution is built from the problem data, the short term memory component attempts to improve it, while avoiding cycles and the middle-term memory component seeks to intensify the search in specified neighborhoods.

The neighborhood $N(S)$ of a solution *S* is defined by all solutions accessible from *S* by applying a move $a \rightarrow b$ to *S*. Indeed, $a \rightarrow b$ is defined as the re-assignment of node *a* to RNC *b*. To evaluate this neighborhood $N(S)$ solution, the gain $G_S(a,b)$ is associated with the move $a \rightarrow b$ and the solution *S* is defined by :

$$G_S(a,b) = \begin{cases} \sum_{i=1,i\neq a}^{n} (h_{ai} + h_{ia}) x_{ib_0} - \sum_{i=1,i\neq a}^{n} (h_{ai} + h_{ia}) x_{ib} + c_{ab} - c_{ab_0} & \text{if } b \neq b_0 \\ M & \text{if not} \end{cases} \quad (8)$$

where:

- h_{ij} refers to the cost of the handover between node i and j;
- b_0 denotes the RNC of node a in solution S, that is, before the application of an $a \rightarrow b$ move;
- x_{ik} takes value 1 if node i is assigned to RNC k, 0 otherwise;
- c_{ik} depicts the cost of linking node i to RNC k;
- M consists of a large arbitrary number.

The short-term memory moves iteratively from one solution to another, applying certain moves and prohibiting a return to the k latest visited solutions. It starts with an initial solution, obtained simply by assigning each node to the closest RNC, according to an Euclidean distance metric. The rationale for this memory component consists of improving the current solution, by decreasing either its cost or penalties.

The middle-term memory component tries to intensify the search in promising regions. It is introduced after the end of the short-term memory component and allows returning to solutions that may have been omitted. It consists mainly of defining intensively searched regions before choosing the types of moves to be applied.

To diversify the search, we use a long-term memory structure in order to guide the search towards regions that have not been explored. This is often done by generating new initial solutions. In this case, a table $n \times m$ (where n is the number of cells and m the number of RNC) counts, for each arc (a,b), the number of times this arc appears in the visited solutions. A new initial solution is generated by choosing, for each cell a, the least visited arc (a,b). Solutions visited during the intensification phase are not taken into account because they result from different types of moves than those applied in short and long-term memory components. From the new initial solution, we start a new search with short and middle term memory mechanisms.

3.2 Simulated annealing approach

Simulated annealing (SA) was introduced by Metropolis et al. [17] and is used to approximate the solution of very large combinatorial optimization problems [13]. Besides the traditional greedy local search techniques, the stochastic properties of the SA algorithm prevent it to get stuck to local minima. On the other hand, in traditional greedy local search, the quality of the final result heavily depends on the initial solution. In contrast, the idea behind SA is to adequately explore the whole solution space early on so that the final solution is insensitive to the starting state [13].

Conversely to a local search algorithm, SA allows for a given optimization problem to accept solutions which deteriorate the cost, even if later, these solutions will be abandoned if they generate no improvements. SA uses randomness to decide whether to reject or accept a solution which deteriorates the cost.

The algorithm starts with an initial feasible solution, which is set as the current solution. Randomly, a neighboring solution from the solution space is chosen, and its cost is compared to that of the current solution. If the cost is improved, this neighbor solution is kept and set as the current solution. Otherwise, this solution is accepted with a probability that is calculated according to the stage the algorithm is in [13].

3.3 Genetic algorithm

Genetic Algorithms (GA) are based on the Darwin's concept of natural selection. They essentially consist of creating a population of candidates and applying probabilistic rules to simulate the evolution of the population. GAs are robust search techniques based on natural selection and genetic mechanisms [6][11][12].

Genetic algorithms (GA) are robust search techniques based on natural selection and genetic production mechanisms [11]. GAs perform a search by evolving a population of candidate solutions through non-deterministic operators and by incrementally improving the individual solutions forming the population using mechanisms inspired from natural genetics and heredity (e.g., selection, crossover and mutation). In many cases, especially with problems characterized by many local optima (graph coloring, travelling salesman, network design problems, etc.), traditional optimization techniques fail to find high quality solutions. GAs can be considered as an efficient and interesting option [11].

GAs [11] are composed of three phases: a phase of creation of an initial population, a phase of alteration of this population by applying various genetic operators on its elements, and finally a phase of evaluation of this population during a certain number of generations. Each generation is supposed to provide new elements better than those of the preceding generation. Intuitively, the more larger is the number of generations, the more refined is the solution. It is hoped that the last generation will contain a good solution, but this solution is not necessarily the optimum.

We have introduced a simple notation to represent cells and switches, and to encode chromosomes and genes. We opted for a non-binary representation of the chromosomes. In this representation, the genes (squares) represent the cells, and the integers they contain represent the switch to which the cell of row i (gene of the i^{th} position) is assigned. Our chromosomes have therefore a length equal to the number of cells in the network n, and the maximal value that a gene can take is equal to the maximal number of switches m. A chromosome represents the set of cells in the cellular mobile network, and the length is the number of cells.

Crossover is a process by which two chosen string genes are interchanged. The crossover of a string pair of length l is performed as follows: a position i is chosen uniformly between 1 and (l-*1*), then two new strings are created by exchanging all values between positions (i+1) and l of each string of the pair considered.

Mutation is the process by which a randomly chosen gene in a chromosome is changed. It is employed to introduce new information into the population and also to prevent the population from becoming saturated with similar chromosomes.

The next generation of chromosomes is generated from present population by selection and reproduction. The *selection* process is based on the fitness of the present population, such that the fitter chromosome contributes more to the reproductive pool; typically this is also done probabilistically.

4. Performance evaluation and numerical results

We submitted our tabu search approach to a series of tests in order to determine its efficiency and sensitivity to different parameters. Thus, we will present a few results, which we compare to those provided by other known heuristics. The most important measurement criterion is the objective function value f (equation 2).

4.1 Experimental settings

This approach was subjected to a series of tests in order to determine its efficiency and sensitivity to different parameters. In the first step, the experiments are executed by supposing that Node Bs are arranged on an hexagonal grid whose length and width are almost equal. The cost of a link between a node and an RNC is proportional to the distance separating both [7]. The call rate γ_i of a node, B*i*, follows a gamma probability distribution with variance equal to one. The call duration at any Node B is exponentially distributed with parameter equal to one [5]. If a node *j* has *k* neighbors, the [0,1] interval is divided into *k+1* sub-intervals by choosing *k* random numbers distributed evenly between 0 and 1. At the end of the service period in node *j*, the call could be either transferred to the i^{th} neighbor (*i=1, …, k)* with a handover probability r_{ij} equal to the length of the i^{th} interval, or ended with a probability equal to the length of the $k+1^{th}$ interval. To find the call volumes and the rates of coherent handovers, the nodes are considered as *M/M/1* queues that form a Jackson network. The incoming rates α_i in Node Bs are obtained by solving the following system:

$$\alpha_i - \sum_{j=1}^{n} \alpha_j r_{ji} = \gamma_i \qquad \text{with} \quad i = 1, \ldots, n \tag{9}$$

If the incoming rate α_i is greater than the service rate, the distribution is rejected and chosen over. The handover rate h_{ij} is defined by:

$$h_{ij} = \lambda_i . \, r_{ij} \tag{10}$$

All the RNCs have the same capacity *M* calculated as follows:

$$M = \frac{1}{m}\left(1 + \frac{K}{100}\right)\sum_{i=1}^{n} \lambda_i \tag{11}$$

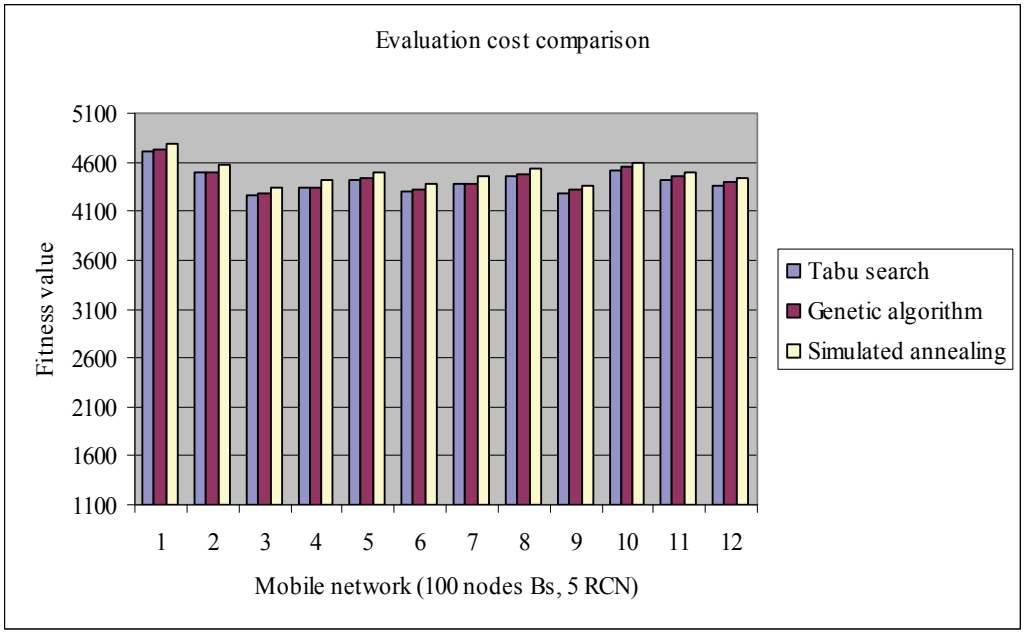
where *K* is uniformly chosen between 10 and 50, which insures a global excess of 10 to 50% of the RNCs' capacity compared to the nodes' call volume.

5.1 Comparison with other heuristics

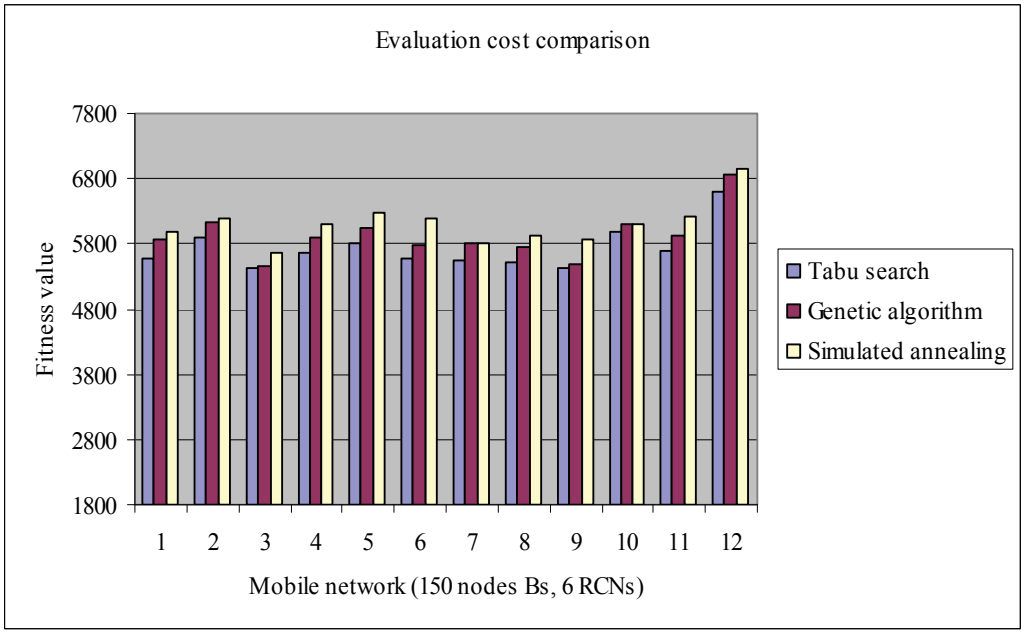
Genetic algorithm (GA) and simulated annealing (SA) are compared with our tabu search solution. In this investigation, a number of Node Bs, varying between 100 and 400, and a number of RNCs fluctuating from 5 to 8 are used, meaning that the search space size spans from 5^{100} to 8^{400}.

The three heuristics consistently find feasible solutions. However, these results inform only about the feasibility of obtained results, without demonstrating whether or not these solutions are among the best. Figure 7 shows the results obtained for 5 different scenarios instances of problems: 5 RNCs and 100 Node Bs, 6 RNCS and 150 Node Bs, 7 RNCs and 200 Node Bs, and 8 RNCs and 400 Node Bs. For each instance, the results of 12 different cases tested show that the evaluation costs represent the average over 100 runs for each algorithm. In each of all the considered series of tests, the tabu search yields an improvement in terms of *f* (2) compared with the other two heuristics. For example, with 7 RNCs and 200 Node Bs, the tabu search solution costs are 3.3% and 4.6% lower than the results generated by GA and SA respectively. Table 2 summarizes the results. Nevertheless, given the initial link, the handover and the annual maintenance costs for large-sized cellular mobile networks this small improvement, in terms of *f* (2), represents a large reduction in costs in the order of millions of dollars over a five-year period. For example, in a cellular network composed of

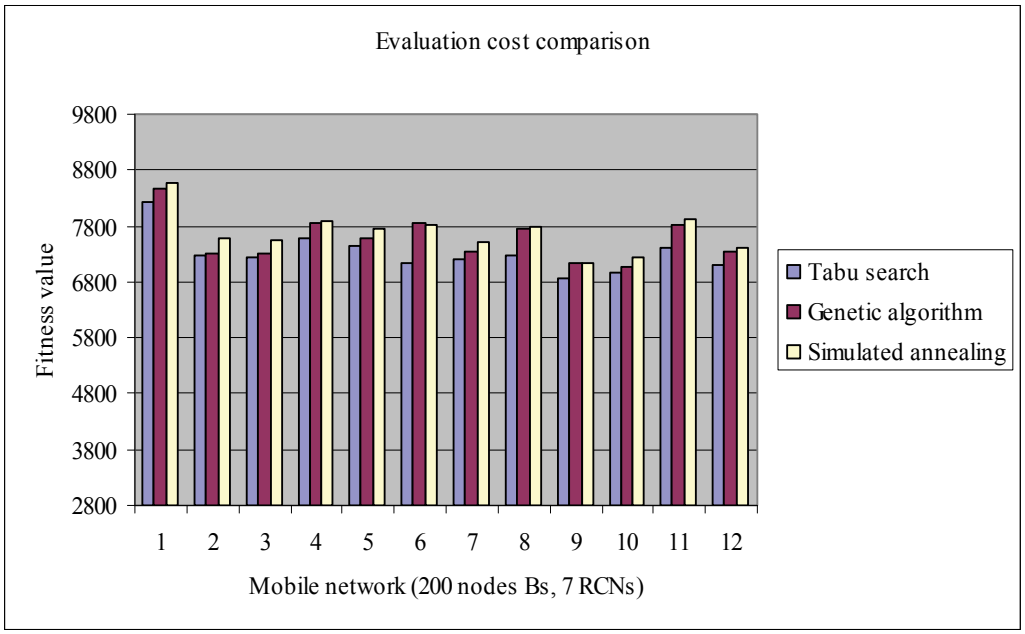
100 Node Bs, with an initial link and handover cost of $ 500,000 for each cell, an improvement of 2% in the cost function represents an approximate saving of $1M over five years. In the case of 500 Node Bs, and an improvement of 5% in the cost function, represents an approximate saving of $12M over 5 years.



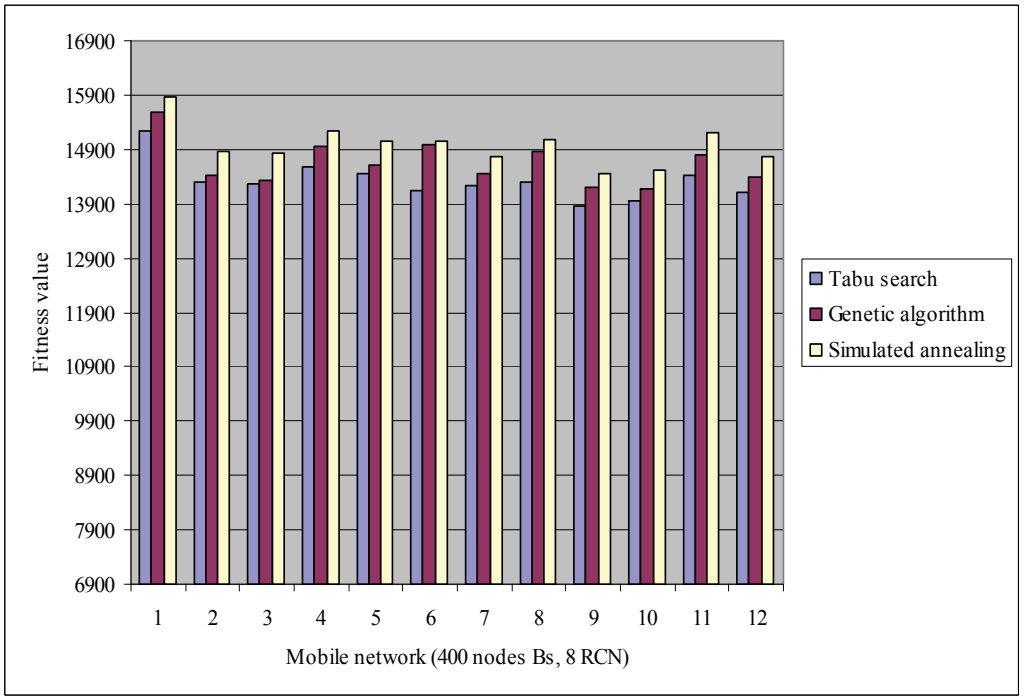
(a) Scenario composed of 5 RNCs and 100 Node Bs



(b) Scenario composed of 6 RNCs, 150 Node Bs



(c) Scenario composed of 7 RNCs, 200 Node Bs



(d) Scenario composed of 8 RNCs, 400 Node Bs

Figure 7. Comparison between tabu search, genetic algorithm and simulated annealing.

	5 RNCs 100 Node Bs	6 RNCs 150 Node Bs	7 RNCs 200 Node Bs	8 RNCs 400 Node Bs
Genetic algorithm	1.6%	2.9%	3.3%	4.3%
Simulated annealing	2.8%	4.1%	4.6%	6.8%

Table 2. Tabu search average improvement rates in terms of $f(2)$.

6. Conclusion

In this paper, we proposed a tabu search approach to design large-scale UMTS mobile networks and to specifically solve the problem of assigning Node Bs to RNCs in cellular mobile networks. Experiments were conducted to measure the quality of solutions provided by this algorithm. This approach was compared against genetic algorithm and simulated annealing.

Computational results obtained confirm the efficiency and the effectiveness of the tabu search to provide better solutions than genetic algorithm and simulated annealing, especially for large-scale cellular mobile networks with a number of Node Bs varying between 100 and 400, and a number of RNCs oscillating between 5 and 8, meaning that the search space size ranges between 5^{100} and 8^{400} and that the average improvement rates are in the order of 2% and 7% respectively. This improvement represents a substantial reduction in maintenance and operations costs, which, for a 5 year period, amount to millions of dollars.

7. References

E. Amaldi, A. Capone, F. Malucelli, "Discrete models and algorithms for the capacitated location problem arising in UMTS network planning", in *Proceedings ACM Dial-M*, 2001, pp. 1–8.

E.Amaldi, A. Capone, F. Malucelli, "Planning UMTS base station location: Optimization models with power control and algorithms", *IEEE Transaction on Wireless Communications*, Vol. 2, No. 5, 2003, pp. 939–952.

R. Beaubrun, S. Pierre, J. Conan, "An efficient Method for Optimizing the Assignment of Cells to MSCs in PCS Networks", *Proceedings 11th International Conference on Wireless Comications.,* Vol. 1, 1999, pp. 259-265.

E. Cayirci, I. Akyildiz, "Optimal location area design to minimize registration signaling traffic in wireless systems", *IEEE Transactions on Mobile Computing,* Vol. 2, No. 1, 2003, pp. 76 – 85.

Y. Fang, I. Chlamtac, Y. Lin, "Modeling PCS Networks Under General Call Holding Time and Cell Residence Time Distributions", *IEEE/ACM Transactions on Networking*, Vol. 5, No. 6, 1997, pp. 893-905.

L. He, N. Mort, "Hybrid genetic algorithms for telecommunications network back-up routing", *BT Technology Journal*, Vol. 18, No. 4, 2000, pp. 42-50.

F. Houeto, S. Pierre, "*Assigning cells to switches in cellular mobile networks using* taboo search", *IEEE Transactions on Systems, Man and Cybernetics, Part B,* Vol. 32, No. 3, 2002, pp. 351 – 356.

B. Gavish, S. Sridhar S., "Economic aspects of configuring *cellular* networks", *Wireless Networks*, Vol. 1 No. 1, 1995, pp. 115-128.

Gavish B., Sridhar, "The Impact of Mobility on Cellular Network Configuration", *Wireless Networks*, Vol. 7 No. 1, 2001, pp. 173-185.

F. Glover, E. Taillard, D. de Werra, "A user's guide to tabu search", *Annals of Operations Research*, Vol. 41, No. 3, 1993, pp. 3-28.

D. Goldberg, "Genetic Algorithms in search, optimization and machines learning", *Addison-Wesley*, 1989.

J. Holland, "Adaptation in natural and artificial systems", *Annals Arbor: The University of Michigan Press*, 1975.

S. Kirkpatrick, D. Gelatt, P. Vecchi, "Optimization by Simulated Annealing", *Science*, Vol. 220, 1983, pp. 671-680.

R. Mathar, T. Niessen, "Optimum positioning of base stations for cellular radio networks", *Wireless Networks*, Vol. 6, No. 6, 2000, pp. 421–428.

A. Merchant, B. Sengupta, "Assignment of Cells to Switches in PCS Networks", *IEEE/ACM Transactions on Networking*, Vol. 3, No. 5, 1995, pp. 521-526.

A. Merchant, B. Sengupta, "Multiway graph partitioning with applications to PCS networks", *IEEE Infocom'94*, Vol. 2, 1994, pp. 593-600.

Metropolis N., Rosenbluth A., Rosenbluth M., Teller A. et Teller E., "Equation for State Calculations by Fast Computing Machines", *Journal of chemical physics*, Vol. 21, 1953, pp. 1087-1092.

M. Nawrocki, M. Dohler, A. H. Aghvami, Eds., *Understanding UMTS Radio Network Modeling, Planning and Automated Optimization: Theory and Practice*, Wiley, 2006.

A. Quintero, S. Pierre, "A Memetic Algorithm for Assigning Cells to Switches in Cellular Mobile Networks", *IEEE Communications Letters*, Vol. 6, No. 11, 2002, pp. 484-486.

A. Quintero, S. Pierre, "Assigning Cells to Switches in Cellular Mobile Networks: A Comparative Study ", *Computer Communications*, Vol. 26, No. 9, 2003, pp 950–960.

D. Reed, "The cost structure of personal communication services", *IEEE communications magazine*, 1993, pp. 102-108.

K. Richardson, "UMTS overview", *Journal IEEE Electronics and Communications Engineering*, Vol. 12, No. 3, 2000, pp. 93–100.

D. Saha, A. Mukherjee, P. Bhattacharjee, "A Simple Heuristic for Assignment of Cell to Switches in a PCS Network", *Wireless Personal Communication*, Vol. 12, 2000, pp. 209-224.

H. Sayoud, K. Takahashi, B. Vaillant, "*Designing communication network topologies using steady-state genetic algorithms*", *IEEE Communications Letters*, Vol. 5, No. 3, 2001, pp. 113–115.

I. Siomina, P. Varbrand, D. Yuan, "*Automated optimization of service coverage and base station antenna configuration in UMTS networks*" *IEEE Wireless Communications*, Vol. 13, No. 6, 2006, pp. 16 – 25.

Y. Wu, S. Pierre, "A New Hybrid Constraint-Based Approach for 3G Network Planning", *IEEE Communications Letters*, Vol. 8, No. 5, 2004, pp. 277-279.

Y. Wu, S. Pierre, "Optimization of Access Network Design in 3G Networks Using a Novel Search Strategy", *Journal of Communications and Networks*, Vol. 7, No. 4, 2005, pp. 471-477.

J. Zhang, J. Yang, M. Aydin, J. Wu, "Mathematical Modeling and Comparisons of Four Heuristic Optimization Algorithms for WCDMA Radio Network Planning", *Proceedings 8th IEEE International Conference on Transparent Networks*, 2006, pp. 253–57.

13

Multiple Tabu Search Algorithm for Solving the Topology Network Design

Kanyapat Watcharasitthiwat, Saravuth Pothiya and Paramote Wardkein
Department of Telecommunication Engineering,
King Mongkut's Institute of Technology Ladkrabang
Thailand

1. Introduction

A communication network can be illustrated by set of nodes (or switches) and links (or arcs) where all nodes are connected by links. The typical communication network structure is composed of two levels. The first one is backbone network and the second one is local access network (LAN). The backbone network dedicate for delivery of information from source to destination (end to end) using its switching nodes. The LAN is typically centralized system that allows users to access hosts or local servers. This chapter focuses only on the backbone network design considered as distributed network.

The advent of low cost devices has led to explosive growth in communication networks. The topology network design is a part of network planning which finds a topology in order to satisfy any constraints. One of the major advantages of distributed network over the centralized system is their flexibility to improve system reliability. The reliability of a system depends not only on the reliability of its nodes and links, but also on the topology of network. A completely connected network has the highest network reliability while the simple loop (ring) network has the lowest network reliability.

Many researchers have studied the designing of the network when considering system reliability as a constraint or an objective. The reliable network design problems state as source-sink and all-terminal network reliability (also known as overall reliability). The problem of optimal topology network design that selects the links connections that either maximizes reliability or minimizes cost can formulate as a combinatorial problem.

This problem is a well known NP-hard problem and very difficult to solve. For such problem, many researchers have studied with enumerative-based and heuristic method. Jan (1993) developed an algorithm using decomposition based on brand and bound to minimize link cost of communication network subject to reliability constraint. Aggarwal et al. (1982) employed greedy heuristic approach to maximize reliability given a cost constraint for networks with different reliability of links and nodes. Pierre et al. (1995) also used simulated annealing to find the optimal design for packet switching networks where considering the delay and capacity, but reliability was not. For the network design, Kumar et al. (1995) developed a genetic algorithm (GA) considering diameter, average distance, and communication network reliability and applied it to four test problems of up to nine nodes. Darren & Smith (1998) presented a GA approach for minimum cost network design problem

with alternative link reliabilities and all-terminal network reliability constraint. Furthermore, Glover et al. (1991) used tabu search (TS) algorithm to choose the topologies of network when considering cost and capacity, but not reliability. Another work of TS algorithm, Beltran & Skorin-Kapov (1994) used TS algorithm to design reliable networks by searching for the least cost spanning two-tree, where the two-tree objective was a coarse surrogate for reliability.

Recently, modern heuristic optimization techniques such as simulated annealing (SA), GA, evolutionary programming (EP), TS algorithm, artificial neural network (ANN), particle swarm optimization (PSO) and ant colony optimization (ACO) have been paid much attentions by many researchers because of their abilities to find an almost global optimal solution. Among of them, the TS algorithm expected as one of the advanced search technique. The TS algorithm is able to escape from local optimal and fast converge to global optimum. However, a conventional TS algorithm might have a problem of reaching the global optimum solution in a reasonable computational time when the initial solution is far a way from the region where optimum solution exists.

Nowadays, the personal computer has more high-speed computation. To solve the large-scale problem, several computers may use for computation at the same time. This method called "parallel searches". On the other hand, the "multiple search" implemented in this chapter is executed by only one personal computer. The multiple searches help to find the promising region where the global optimum solution exists. This idea applies for improvement the performance of a conventional TS algorithm.

Pothiya et al. (2006) developed the multiple tabu search (MTS) algorithm and applied this algorithm to design the optimal fuzzy logic PI controller. After that, MTS applied to solve the economic dispatch problem with generator constraint (Pothiya et al., 2008). The MTS is the execution of individual TS algorithm simultaneously by only single personal computer. The MTS introduces the additional salient mechanisms for improvement of search process, i.e. initialization, adaptive searches, multiple searches, replacing and restarting process. The feasibility study of the MTS for this chapter demonstrated for solving the topology network design when considering both reliability and cost. The optimized results by the MTS compared to those obtained by the conventional approaches such as GA, TS and ACO in terms of solution quality and computational efficiency.

The remainders of this chapter organize as follows. Section 2 describes the problem formulation, assumptions of system, and reliability calculation. Section 3 presents the principle of MTS algorithm for solving the topology network design. Section 4 illustrates the examples and simulation results. Finally, Section 5 contains the discussion and conclusion.

2. Statement of the problem

Notations

L	Set of possible link
l_{ij}	Option of each link
d_{ij}	Distance between node i and node j
N	Set of given node
n	Number of node
$p(l_k)$	Reliability of link option

$c(l_k)$ Unit cost of link option
x Architecture of network design
$C(x)$ Total cost of network design
$R(x)$ Reliability of network design
R_o Minimum network reliability constraint

2.1 Problem formulation

In the source-sink and all-terminal reliability network design problem, there are a set of N nodes with specified topology, which can be found from real world networks or will be interpreted as Euclidean distance between coordinates on a plane. It is noted that this distance is not an existing distance. It only represents some cost of connection between two nodes regardless of type of connection. The network nodes are assumed to be fully reliable or assumed not to fail under any circumstances. There are a set of L links, which connect all nodes in N. In this problem, it is assumed that every possible links are included in L, i.e., a fully connected network. Hence, for any (n_i, n_j) pair of elements of N there exists the possibility of l_{ij} element of L such that l_{ij} connects n_i and n_j. In addition, it is assumed that a link is bi-directional if l_{ij} is turned on it lets n_i communicate with n_j and vice versa. It is also assumed that there is only one link per location. Therefore, all links fail independently and repairing link is required if any link fails. The total number of possible links in single design is:

$$|L| = \frac{|N| \cdot |N-1|}{2} \tag{1}$$

Generally, one link can possibly have more than two states. Thus a candidate solution, x, to this problem then consists of a selected number of links $l_{ij} = k$ where k is the level at which those links connect nodes n_i and n_j. The mathematical formulation for this problem when minimizing cost is subject to a minimum network reliability constraint is:

$$\text{Minimize } C(x) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{i,j} \cdot l_{i,j} \cdot d_{ij} \tag{2}$$

$$\text{Subject to } \quad R(x) \geq R_0 \tag{3}$$

The cost of a specific architecture x is given by $C(x)$ and the reliability of x is given by $R(x)$. The problem find an x so that $R(x) \geq R_0$ base on the following assumptions**:**

1. The locations of all the network nodes are given.
2. The cost c_{ij} and the operation probability p_{ij} of each link (i, j) are fixed.
3. Each link is bi-directional.
4. No redundant link is allowed in the network.

2.2 Reliability calculation

The problem of calculating or estimating the reliability of a network is another active area of research related to the economic network design problem. There are two main approaches

the exact calculation through analytic methods (Ball & Van-Slyke 1977, Jan 1993) and the estimation calculation through Monte Carlo simulation (Fishman 1986, Ramirez-Marquez & Coit 2005). For the all-terminal network reliability problem, efficient simulation is difficult because these methods generally lose efficiency as a network approaches a fully connected state. There are also upper and lower bound expressions for network reliability (Shinmori & Ishii 1995), however these are too loose to be effective surrogates in the all-terminal design process. Furthermore, many bounding procedures and improved efficiency simulations depend on the assumption that all arcs have the same reliability, which is relaxed in this research. Therefore in this chapter, either the network reliability was calculated exactly using a backtracking procedure or a classic Monte Carlo procedure. Due to the computationally tractable size, a backtracking algorithm is used to correctly calculate the system reliability, $R(x)$, for the problems in this chapter. The outline of the backtracking algorithm is given as follow:

Step 0: (Initialization). Mark all links as free; create a stack which is initially empty.

Step 1: (Generate modified cut-set)

(a) Find a set of free links that together with all inoperative links will form a network-cut.

(b) Mark all the links found in 1(a) inoperative and add them to the stack.

(c) The stack now represents a modified cut-set; add its probability into a cumulative sum.

Step 2: (Backtracking)

(a) If the stack is empty, end.

(b) Take a link off the top of the stack.

(c) If the link is inoperative and it is operative a spanning tree of operative links exists, then mark it free and go to 2(a)

(d) If the link is inoperative and the condition tested in 2(c) does not hold, then mark it operative, put it back on the stack and go to Step 1.

(e) If the link is operative, then mark it free and go to 2(a).

It should be mentioned that the algorithm above is for all-terminal reliability and needs to be modified for use in a source-sink design problem as given below:

Step 0: (Initialization). Mark all links as free; create a stack that is initially empty.

Step 1: (Generate modified cut-set)

(a) Find a set of free links that together with all inoperative links will form a source-sink cut.

(b) Mark all the links found in 1(a) inoperative and add them to the stack.

(c) The stack now represents a modified cut-set; add its probability to a cumulative sum.

Step 2: (Backtracking)

(a) If the stack is empty, end.

(b) Take a link off the top of the stack.

(c) If the link is inoperative and if when made operative, a path from the source to the sink exists, then mark it free and go to 2(a).

(d) If the link is inoperative and the condition tested in 2(c) does not hold, then mark it operative, put it back on the stack and go to Step 1.

(e) If the link is operative, then mark it free and go to 2(a).

For larger networks, Monte Carlo simulation is used to accurately estimate network reliability. The network is simulated t times given the design and the link reliabilities:
Initialize $i = 0$, $c = 0$
Step 0: While $i < t$ Repeat.
Step 1: Randomly generate network.
(a) $i = i + 1$
Step 2: Check to see if the network forms a spanning tree.
(a) If the network forms a spanning tree then $c = c + 1$ go to Step 0.
(b) If the network does not form a spanning tree go to Step 0.
Step 3: $R(x) = c / t$

When the need to simulate a network's reliability arises, other issues become important. One of these issues is whether or not the estimator is biased. The other issue is the variance of the estimate. Every referenced Monte Carlo technique is an unbiased estimator where the variance of the Monte Carlo method described above is:

$$\text{Var}\ (\ R(x)\ ) = \frac{R(x)(1-R(x))}{t} \tag{4}$$

To get a more accurate reliability estimate, t must be high value.
Note: Var is the variance of variable.

3. Multiple tabu search algorithm

The MTS algorithm is the execution of individual TS algorithm at the same time by a single personal computer. Here, the individual TS algorithm and the MTS algorithm are explained.

3.1 Principle of tabu search

1) Overview

In general terms, a conventional TS algorithm is an iterative search that starts from an initial feasible solution and attempts to determine a better solution in the manner of a hill-climbing algorithm. The TS has a flexible memory to keep the information about the past steps of the search. The TS uses the past search to create and exploit the better solutions (Glover 1989, Glover 1990). The main two components of TS algorithm are the tabu list (TL) restrictions and the aspiration criterion (AC).

2) Tabu list restrictions

In order to prevent cycling, repeated search at the same solution, a TL is introduced. The TL stores a set of the tabu (prohibition) moves that can not be applied to the current solution. The moves stored in TL called *tabu restrictions* and used for decreasing the possibility of cycling because it prevents returning in a certain number of iterations to a solution visited recently.

In this chapter, the size of TL is $n \times 3$ (row $\times$ column), n is a number of neighborhoods around current solution. In the TL, the first column is used for storing the moves, the second column is the frequency of a move direction, and the last column is the recency (time to keep solutions) of a move (Bland & Dawson 1991).

3) Aspiration criterion

Another key issue of TS algorithm arises when all moves under consideration have been found to be tabued. The tabu status of a move is not absolute, but it can be overruled if

certain conditions are met and expressed in the form of AC. If appropriate aspiration criterion is satisfied, the move will be accepted in spite of the tabu classification. Roughly speaking, AC is designed to override tabu status if a move is 'good enough' (Bland & Dawson 1991).

4) Stopping criterion

There are several possible conditions for stop searching. Here, the stopping search is used if any of the following two conditions are satisfied: first, the accuracy of the best solution is lower than the expected value, and the second, the maximum allowable number of iterations is reached.

5) General tabu search algorithm

To solve a combinatorial optimization problem by tabu search, the basic idea is to choose randomly a feasible solution and attempt to find a best neighbor to current solution. A move to this neighbor is performed if either it does not belong to the TL or, it passes the AC test. During these procedures, the best solution is always updated and stored aside until the stopping criterion is satisfied. The following notations are used through the description of the TS algorithm for a general combinatorial optimization problem:

X : the set of feasible solutions
x : the current solution, $x \in X$
x_b : the best solution reached
x_{nb} : the best solution among of trial solutions
$E(x)$: the objective function of solution x
$N(x)$: the set of neighborhood of $x \in X$
TL : tabu list
AC : aspiration criterion

The procedure of TS algorithm is described as follows:

Step 0: Set TL as empty and AC to be zero.

Step 1: Set iteration counter $k = 0$. Select an initial solution $x \in X$, and set $x_b = x$.

Step 2: Generate a set of trial solutions neighborhood of x . Let x_{nb} as the best trial solution.

Step 3: If $E(x_{nb}) > E(x_b)$, go to Step 4, else set the best solution $x_b = x_{nb}$ and go to Step 4.

Step 4: Perform the tabu test. If x_{nb} is NOT in the TL , then accept it as a current solution, set $x = x_{nb}$, and update the TL and AC and go to Step 6, else go to Step 5.

Step 5: Perform the AC test. If satisfied, then override the tabu state, set $x = x_{nb}$, update the AC .

Step 6: Perform the termination test. If the stopping criterion is satisfied then stop, else set $k = k + 1$ and go to Step 2.

3.2 Multiple tabu search for solving topology network design

Although the TS algorithm is able to escape from local optimal and fast converge to global optimum. It might have a problem with reaching the global optimum solution in a reasonable computation time when an initial solution is far away from the region where the optimum solution exists. The convergence speed of TS algorithm depends on an initial solution. The convergence speed can be improved by introducing a multiple structure into the algorithm.

The MTS algorithm uses several initial solutions to increase the probability of reaching the region where the optimum solution exists. The procedure of the MTS algorithm is depicted in Fig. 1, which consists of several independent conventional TS algorithms (TS#1, TS#2,

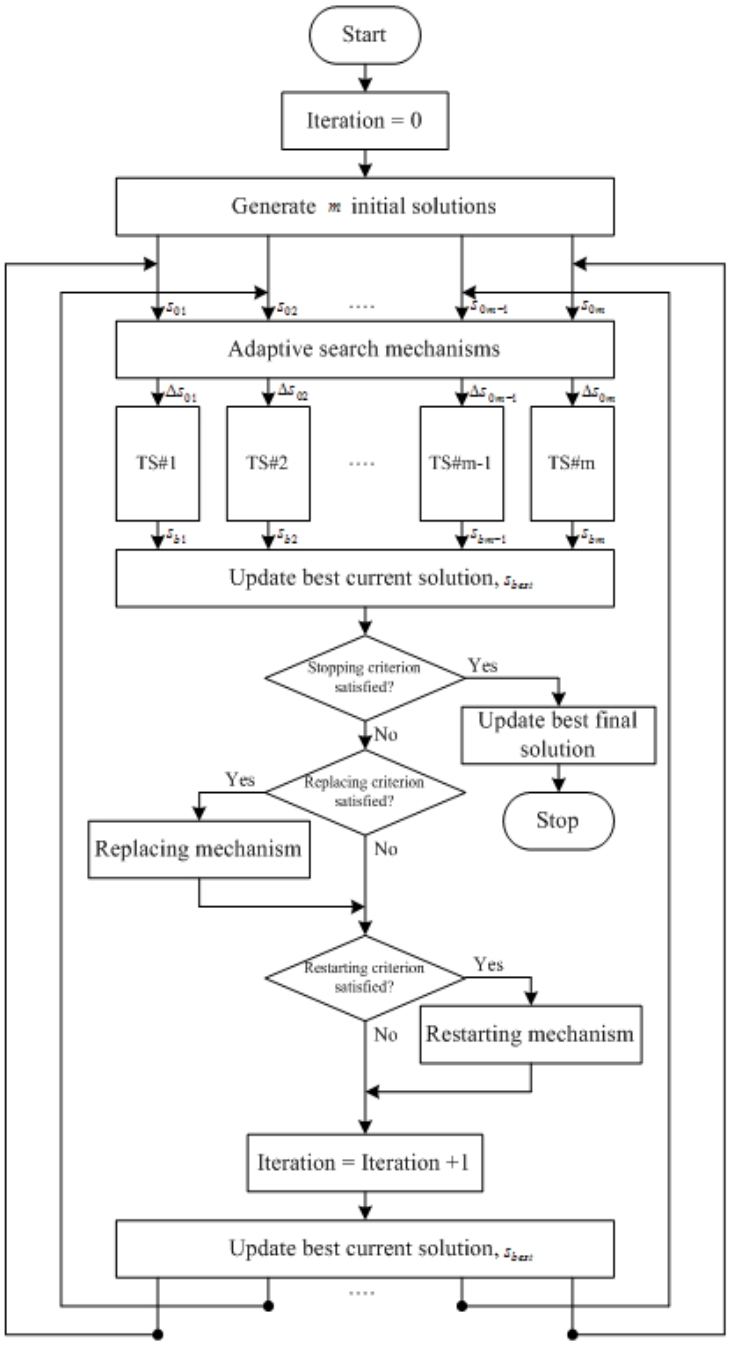


Fig. 1 Procedure of Multiple Tabu Search algorithm

..,TS#m). Furthermore, the additional mechanisms namely, *initialization*, *adaptive searches, multiple searches*, *replacing* and *restarting process* help to improve the search process in terms of both solution quality and computational time. The additional mechanisms are explained as follows:

1) Initialization

To improve searching, the MTS algorithm starts to search from several initial solutions which are different from the TS algorithm. In fact, the starting with several initial solutions has the probability of reaching the optimum solution higher than the single initial solution. This mechanism helps MTS algorithm to converge quickly to the global optimum solution.

In the initialization process, several initial solutions are created randomly for individual TS algorithm. In this chapter, the structure of a solution for topology network problem is composed of a set of links. Therefore, the initial solutions of individual TS algorithm at iteration #0 can be represented as the matrix of X_i^0, $i=1,\ldots,m$ where m is the number of multiple TS algorithms. Note that it is very important to create a set of solution satisfying the equality and inequality constraints.

2) Adaptive searches

The step size is the range of variance at current solution which is the important factor for search process. Accordingly, the step size should be chosen appropriately. In general, this value is fixed. Low value of the step size can increase the accuracy of solution. But it takes a long computational time. On the other hand, high value of the step size is used for decreasing the computational time. But the searched result may not reach the global optimum. Consequently, the adaptive search mechanism has been developed to adjust suitably the step size during the search process. This mechanism helps to increase the computational speed and the accuracy of solution.

3) Multiple searches

Nowadays, the personal microcomputer has high speed computation. To solve the large-scale problem, several computers may be used at the same time. This method is called *parallel searches*. For *multiple searches*, they are executed by only one personal microcomputer. The multiple searches help to find the promising region where the global optimum solution exists.

The MTS algorithm uses the multiple searches mechanism to enhance its capacity. The multiple searches mechanism is executed by using only a personal microcomputer. Each step of searching for finding the better solution is used in the procedure of TS algorithm which is given in section 3.1 (5). The sequence of execution starts from TS#1 to TS#m. Fig. 2 illustrates the sequence of execution for finding better solutions.

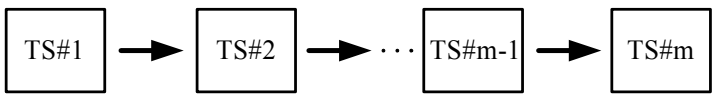


Fig. 2 Sequence of execution

4) Replacing

After the search process satisfies with the condition for replacing, all independent TS algorithms are stopped. The condition for replacing depends on the size of problem. The experiment can be found the appropriate value. The replacing mechanism is used for

comparison and exchanging the solutions which are found by these TS. Then, the replacing mechanism generates the best initial solutions for the next search. Like the crossover of GA, the MTS algorithm performs with replacing mechanism for improvement the solution. Here, the replacing is set to apply to each individual TS every 30 iterations. As a result, new solutions are generated for the next iteration.

5) Restarting process

When the search is stroke on the local solution for a long time and the procedure of TS algorithm can not escape from the local solution. The restarting process is applied to keep searching and finding a new solution. The restarting mechanism is applied when the search is stroke on the local solution for 20 iterations and the procedure of TS algorithm can not escape from local solution. The procedure of restarting mechanism is almost the same as the initialization mechanism.

4. Test problems and results

Three test problems were studied and each exhibits a different aspect of the MTS approach. These four link types, shown in Table 1 were used for all test problems. To examine the performance of the proposed MTS method, it is then compared the results with those of GA, TS and ACO. The simulation results shown in this chapter are simulated using MatLab® program on a Pentium 4, 2.66 GHz 512 DRAM personal computer.

Connection Type	Reliability	Cost ($)/Unit Distance
0 (not connected)	0.00	0
1	0.85	8
2	0.90	10
3	0.95	14

Table 1 Link Unit Costs and Corresponding Reliabilities

4.1 Application test problems

1) Test Problem 1: Five nodes network

A communication network has five nodes with multiple levels as show in Fig. 3. The problem is based on Jan (1993) but expanded by changing the links from a simple on/off state to one of four possible states using link costs as distances and using the unit costs and reliabilities shown in Table 1. Since this problem has $k = 4$ levels its search space size is $4^{(5\times4)/2} = 1{,}048{,}576$. This problem was considered at six different system reliability constraints and the backtracking algorithm is employed to calculate $R(x)$ exactly. The cost matrix of links is given as follow:

$$[l_{ij}] = \begin{bmatrix} - & 32 & 54 & 62 & 25 \\ & - & 34 & 58 & 45 \\ & & - & 36 & 52 \\ & & & - & 29 \\ & & & & - \end{bmatrix} \tag{5}$$

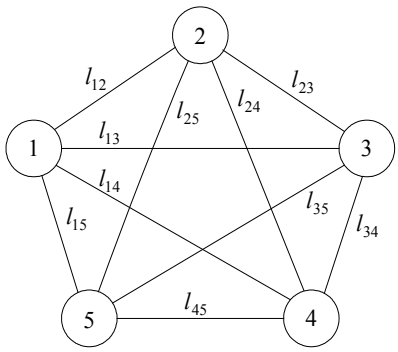


Fig. 3 Five nodes communication network system

2) Test Problem 2: Source-Sink network

This problem shows that the flexibility of the approach by considering economic design of 18 links source-sink network as shown in Fig. 4. This problem is taken from the literature (Jan 1993) and has 6.9×10^{10} possible architectures, thus precluding enumeration to identify the optimal design. The distance matrix for this problem appears in Table 2. This problem was considered at six different system reliability constraints and uses the backtracking algorithm to calculate the exact value of $R(x)$.

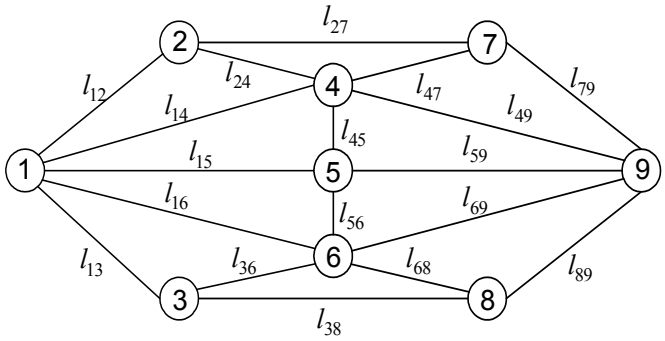


Fig. 4 Source-Sink network (Source node is 1 and Sink node is 9)

	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	58	63	60	63	58	-	-	-
2		42	-	-	-	60	-	-
3			20	-	-	42	-	63
4				20	-	-	-	60
5					42	-	42	63
6						-	60	-
7							-	58
8								58

Table 2 Distance matrix for Source-Sink network

3) Test Problem 3: 19 Districts in Bangkok, Thailand

To demonstrate the applicability of this work on a realistic application, the network of 19 districts in Bangkok is therefore applied for an example of scaled-up problem. This 19 node problem was constructed by selecting districts in Bangkok as seen in Fig. 5 and computing the Euclidean distances between them using their coordinates as shown in Table 3 and using the four links choices from Table 1. The search space of this problem is 8.959×10^{102} . Besides the scale-up issue, the difference between this problem and the other design problems is that this problem illustrates the flexibility of the MTS by reversing the constraint and the objective function. This problem was considered at six different system reliability constraints and uses the Monte Carlo simulation is used to accurately estimate network reliability.

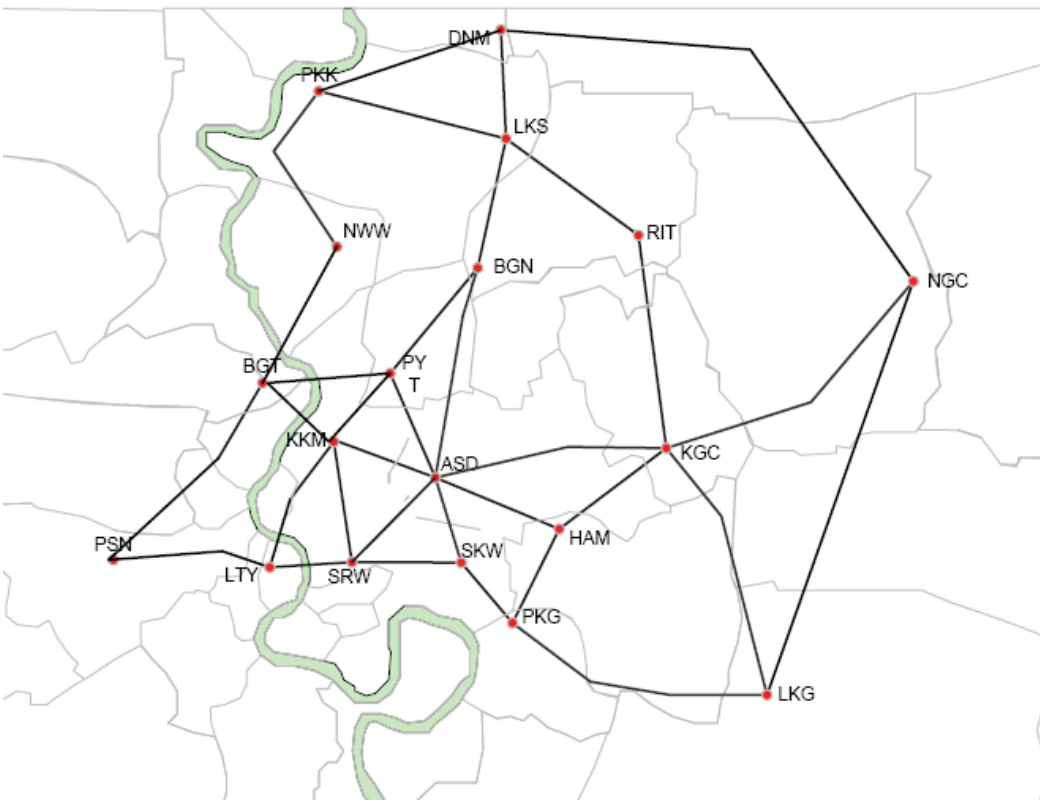


Fig. 5 Network of 19 districts in Bangkok, Thailand

4.2 Simulation results

Four methods, which are MTS, GA, TS and ACO, are employed to examine three test problems with six different system reliability constraints. To perform 30 trials, the simulation results in terms of maximum, average and minimum cost, the standard deviation, the average computational time, and percentage of get near optimum solution for test problem 1, 2 and 3 are presented in Table 4, 5 and 6, respectively. Obviously, the

proposed MTS approach always provided better solution than other methods, thus resulting in the higher quality solution.

In order to demonstrate the efficiency of the proposed MTS method, the distribution outlines of the best solution of each trial are considered. Fig. 6 and 7 demonstrate the distribution outlines of best solution of each trail for test problem 1 and 2, respectively. Most of the costs obtained by the proposed MTS method were lower than those of the compared methods, that the efficiency of the proposed MTS method is superior to other methods.

	LKS	PKK	NWW	RIT	BGN	NGC	BGT	PYT	KKM	KGC	ASD	PSN	LTY	SRW	SKW	HAM	PKG	LKG
DNM	8.10	19.40	27.88	17.50	12.80	45.41	29.92	20.30	30.20	25.90	36.59	48.62	36.61	38.90	40.47	31.90	38.00	66.06
LKS		11.30	19.78	9.40	4.70	45.28	21.82	12.20	22.10	17.80	28.49	40.52	28.51	30.80	32.37	23.80	29.90	41.75
PKK		-	8.48	20.70	16.00	62.01	25.80	25.58	26.08	29.10	39.79	44.50	32.49	32.18	43.67	45.49	49.37	49.75
NWW			-	29.18	24.48	70.49	17.32	17.10	17.60	19.35	25.10	36.02	24.01	23.70	28.98	30.80	34.68	54.27
RIT				-	14.10	35.88	31.22	21.60	31.50	8.40	20.10	49.92	37.91	30.00	26.00	14.20	20.30	32.35
BGN					-	52.96	17.12	7.50	17.40	25.48	23.79	35.82	23.81	34.69	27.67	29.49	33.37	52.96
NGC						-	56.80	47.18	57.08	27.48	39.18	65.74	55.88	50.08	43.06	33.48	39.58	20.65
BGT							-	9.62	5.32	28.97	17.62	18.70	11.96	17.76	12.92	23.32	18.62	38.21
PYT								-	9.90	19.35	8.00	28.32	16.31	16.00	11.88	13.70	17.58	37.17
KKM									-	20.90	9.20	16.27	6.41	6.10	7.60	14.90	13.30	32.89
KGC										-	11.70	38.26	28.40	22.60	12.10	6.00	12.10	23.95
ASD											-	26.56	16.70	10.90	3.88	5.70	9.58	29.17
PSN												-	9.86	15.66	24.96	36.76	30.66	50.25
LTY													-	5.80	15.10	26.90	20.80	40.39
SRW														-	9.30	21.10	15.00	34.59
SKW															-	11.80	5.70	25.29
HAM																-	6.10	25.69
PKG																	-	19.59

DNM	Donmuang	LKS	Laksi	PKK	Pakkret
NWW	Ngamwongwan	RIT	Ram-intra	BGN	Bangken
NGC	Nongchok	BGT	Bangplad	PYT	Paholyothin
KKM	Krungkasem	KGC	Klongchan	ASD	Asok Dindeang
PSN	Pasicharoen	LTY	Latya	SRW	Surawong
SKM	Sukhumwit	HAM	Huamark	PKG	Phrakanong
LKG	Ladkrabang				

Table 3 Distance matrix of 19 districts in Bangkok, Thailand

Reliability	Optimal Cost	Configuration	Algorithm	Max. Cost	Average Cost	Min. Cost	Standard Deviation	% Get Optimal	CPU Time
0.999	5498	3323323333	GA	5498	5498	5498	0.00000	100	12.49
			TSA	5498	5498	5498	0.00000	100	9.93
			ACO	5498	5498	5498	0.00000	100	7.11
			MTS	**5498**	**5498**	**5498**	**0.00000**	**100**	**3.15**
0.995	4250	3303222313	GA	4250	4250	4250	0.00000	100	47.21
			TSA	4250	4250	4250	0.00000	100	43.65
			ACO	4250	4250	4250	0.00000	100	35.50
			MTS	**4250**	**4250**	**4250**	**0.00000**	**100**	**12.11**
0.990	3638	3203312303	GA	3900	3760	3638	88.05423	13	57.12
			TSA	3886	3700	3638	68.02699	17	55.53
			ACO	3686	3654	3638	21.00489	50	44.89
			MTS	**3638**	**3638**	**3638**	**0.00000**	**100**	**10.24**
0.950	2184	3003300303	GA	2692	2481	2184	121.65421	56	57.35
			TSA	2622	2305	2184	98.014345	80	31.52
			ACO	2444	2193	2184	47.469288	97	22.11
			MTS	**2184**	**2184**	**2184**	**0.00000**	**100**	**6.53**
0.900	1904	3003200203	GA	2402	1982	1904	126.28258	43	47.55
			TSA	2124	1937	1904	87.234497	93	17.51
			ACO	1904	1904	1904	0.00000	100	14.27
			MTS	**1904**	**1904**	**1904**	**0.00000**	**100**	**2.36**
0.850	1724	2003300102	GA	1764	1734	1724	10.061878	90	21.12
			TSA	1736	1726	1724	4.290594	100	1536.
			ACO	1732	1727	1724	3.835706	100	8.11
			MTS	**1724**	**1724**	**1724**	**0.00000**	**100**	**1.10**

Table 4 Summary results of five nodes problem (performed 30 trails)

Reliability	Optimal Cost	Configuration	Algorithm	Max. Cost	Average Cost	Min. Cost	Standard Deviation	% Get Optimal	CPU Time
0.999	6008	033300000001013330	GA	6008	6008	6008	0.00000	100	63.41
			TSA	6008	6008	6008	0.00000	100	55.48
			ACO	6008	6008	6008	0.00000	100	26.42
			MTS	**6008**	**6008**	**6008**	**0.00000**	**100**	**15.23**
0.995	4464	022200000000003330	GA	5382	4795	4464	232.843107	13	112.42
			TSA	4970	4681	4464	200.417828	36	105.20
			ACO	4596	4506	4464	56.766673	63	50.87
			MTS	**4464**	**4464**	**4464**	**0.00000**	**100**	**10.24**
0.990	4074	013100000000001330	GA	5212	4190	4074	265.212310	76	72.66
			TSA	4952	4189	4074	251.683618	80	69.75
			ACO	4086	4078	4074	5.881590	100	42.27
			MTS	**4074**	**4074**	**4074**	**0.00000**	**100**	**18.42**
0.950	2826	002100000000000330	GA	3446	2980	2826	164.221423	30	106.43
			TSA	3112	2922	2826	147.749415	53	93.21
			ACO	3248	2900	2826	130.371018	70	72.68
			MTS	**2826**	**2826**	**2826**	**0.00000**	**100**	**22.56**
0.900	2328	011000000000001300	GA	2648	2386	2328	75.218902	57	90.23
			TSA	2530	2374	2328	47.981001	60	79.92
			ACO	2394	2337	2328	22.816509	87	49.98
			MTS	**2328**	**2328**	**2328**	**0.00000**	**100**	**21.23**
0.850	1990	002000001000000220	GA	2312	2074	1990	97.273386	50	65.98
			TSA	2335	2104	1990	105.07909	48	63.02
			ACO	2180	2008	1990	70.477535	93	61.39
			MTS	**1990**	**1990**	**1990**	**0.00000**	**100**	**25.14**

Table 5 Summary results of source-sink nodes problem (performed 30 trails)

Reliability	Algorithm	Max. Cost (฿)	Average Cost (฿)	Min.Cost (฿)	Standard Deviation
0.999	GA	10,364,282	10,361,136	10,359,058	1522.41
	TSA	10,361,728	10,359,303	10,356,430	1305.12
	ACO	10,358,095	10,357,185	10,356,121	567.75
	MTS	**10,343,501**	**10,331,632**	**10,324,514**	**315.54**
0.995	GA	8,274,131	8,271,620	8,269,961	1215.39
	TSA	8,272,093	8,270,157	8,267,863	1041.90
	ACO	8,269,192	8,268,466	8,267,616	453.25
	MTS	**8,257,863**	**8,251,491**	**8,245,361**	**251.62**
0.990	GA	7,022,953	7,020,822	7,019,414	1031.60
	TSA	7,021,223	7,019,579	7,017,633	884.35
	ACO	7,018,761	7,018,144	7,017,423	384.71
	MTS	**6,899,623**	**6,898,631**	**6,894,144**	**160.55**
0.950	GA	4,975,827	4,974,316	4,973,319	730.90
	TSA	4,974,601	4,973,436	4,972,057	626.56
	ACO	4,972,856	4,972,419	4,971,908	272.57
	MTS	**4,954,615**	**4,950,241**	**4,942,416**	**121.19**
0.900	GA	4,443,038	4,441,690	4,440,799	652.54
	TSA	4,441,944	4,440,904	4,439,672	559.47
	ACO	4,440,586	4,439,996	4,439,540	253.39
	MTS	**4,425,619**	**4,418,629**	**4,412,612**	**110.82**
0.850	GA	4,021,989	4,020,768	4,019,962	622.54
	TSA	4,020,998	4,020,056	4,018,942	590.79
	ACO	4,019,988	4,019,234	4,018,822	280.32
	MTS	**3,892,512**	**3,889,415**	**3,867,642**	**135.46**

Note: 1 $US = 35 ฿

Table 6. Summary results of 19 districts in Bangkok, Thailand (performed 30 trails)

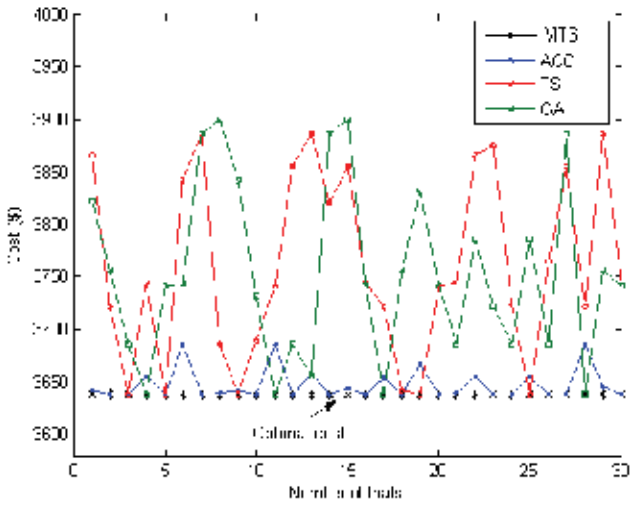


Fig. 6 Distribution of cost for test problem 1 ($R_0 = 0.990$)

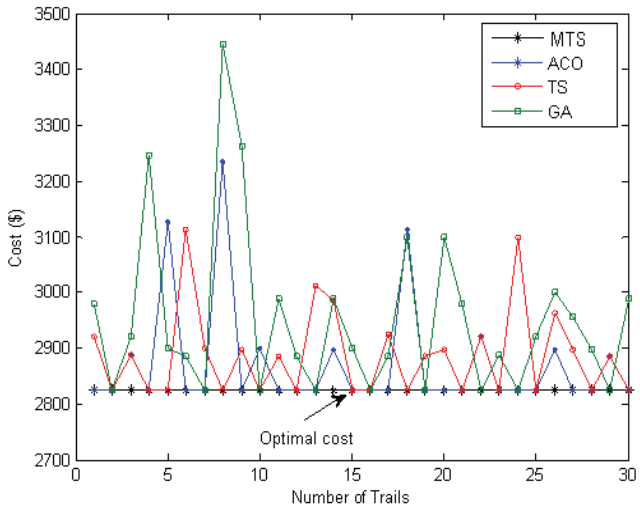


Fig. 7. Distribution of cost for test problem 2 ($R_0 = 0.950$)

5. Conclusion and discussion

In this chapter, the MTS method has been applied to solve the topology network design problem with considering both economics and reliability. The MTS algorithm shows superior features such as high-quality solution, stable convergence characteristic, and good computation efficiency. The studied results confirm that the proposed MTS are indeed capable of obtaining higher quality solution efficiently, convergence characteristic and computation efficiency in comparison with GA, TS and ASO methods.

6. References

Aggarwal, K. K.; Chopra, Y. C. & Bajwa, J. S. (1982). Reliability evaluation by network decomposition : *IEEE Transactions on Reliability*, Vol. R-31, pp. 355-358.

Ball, M. & Van Slyke, R. M. (1977). Backtracking algorithms for network reliability analysis: *Annals of Discrete Mathematics*, Vol. 1, pp. 49-64.

Bell, J. & McMullen, P. (2004). Ant colony optimization techniques for the vehicle routing problem: *Advanced Engineering Informatics*, Vol. 18, pp. 41-48.

Beltran, H. F. & Skorin-Kapov, D. (1994). On minimum cost isolated failure immune networks: *Telecomm. Sys*, Vol. 3, pp. 183-200.

Bland, J. A. & Dawson, G. P. (1991). Tabu Search and Design Optimization: *Computer - aided design*, Vol. 23, No. 3, pp. 195-201.

Chelouah, R. & Siarray, P. (2000). Tabu Search applied to global optimization: *European Journal of Operational Research,*. Vol. 123, pp. 256-270.

Coloni, A; Dorigo, M. & Maniezzo, V. (1991). Distributed optimization by ant colonies: *Processding of Eur. Conf. Artificial Life*, pp. 134-142.

Darren, L. D. & Smith, A. E. (1998). Economic Design of Reliable Networks: *IEEE Transactions on Economic of Reliability Engineering*, July.

Dorigo, M.; Maniezzo, V. & Colorni, A. (1996). Ant system: Optimization by a colony of cooperative agents: *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, Vol. 26, No. 1, pp. 29-41.

Dorigo, M. & and Gambardella, L. M. (1997). Ant colonies for the traveling saleman problem: *Biosystems*, Vol. 43, pp. 73-81.

Dorigo, M. & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling saleman problem: *IEEE Trans. Evolut. Comput.*, Vol. 1, No. 1, pp. 53-66.

Fanni, A. et al., (1999). Tabu Search metaheuristics for electromagnetic problems optimization in continuous domain: IEEE Trans. On Magnet. Vol. 34, no. 3, pp. 1694-1697.

Fishman, G. S. (1986). A comparison of four Monte Carlo method for estimating the probability of s-t connectedness: *IEEE Transactions on Reliability,* Vol. 35, No. 2, pp. 145-155.

Gambardella, L. M.; Taillard, E. & Dorigo, M. (1999). Ant colonies for the quadratic assignment problem*: J. Oper. Res. Soc.*, Vol. 50, pp. 167-176.

Glover, F. (1989). Tabu Search Part I: *ORSA J. Comput*, Vol. 1 No. 3, pp. 190-206.

Glover, F. (1990). Tabu Search Part II: *ORSA J. Comput*, Vol. 2 No. 1, pp. 4-32.

Glover, F.; Lee, M. & Ryan, J. (1991). Least-cost network topology design for a new service: An application of a tabu search: *Ann. Oper. Res*., Vol. 33, pp. 351-362.

Ho, S. L.; Yang, S.; Wong, H. C.; Cheng, K.W.E. & Ni, G. (2005). An Improved Ant Colony Optimization Algorithm and Its Application to Electromagnetic Devices Designs: *IEEE Transactions on Magnetics,* Vol. 41, No. 5, pp. 1764-1767.

Jan, R. H (1993). Design of reliable networks: *Computers and Operations Research*, Vol. 20, No. 1, pp. 25-34.

Jan, R. H.; Hwang, F. J. & Chen, S. T. (1993). Topology optimization of a communication network subject to a reliability constraint: *IEEE Transactions on Reliability*, Vol. 42, No. 1, pp. 63-70.

Kumar, A.; Pathak, R. M. ; Gupta Y. P. & Parsaei, H. R. (1995). A genetic algorithm for distributed system topology design: *Comp. Ind. Eng*., Vol. 28, pp. 659-670.

Mathur, M.; Karale, S. B.; Priyee, S.; Jayaraman, V. K. & Kulkarni, B. D. (2000). Ant colony approach to continuous function optimization: *Ind. Eng. Chem. Res.*, Vol. 39, pp. 3814-3822.

Pierre, S; Hyppolite, M. A. ; Bourjolly, J.M. & Dioume, O. (1995). Topology design of computer communication network using simulated annealing: *Eng. Applicat. Artificial Intell*., Vol. 8, pp. 61-69.

Pothiya, S.; Ngamroo, I.; Runggeratigul, S. & Prinya Tantaswadi, P. (2006). Design of Optimal Fuzzy Logic based PI Controller using Multiple Tabu Search Algorithm for Load Frequency Control: *International Journal of Control, Automation, and Systems*, Vol. 4, No. 2, pp. 155-164.

Pothiya, S.; Ngamroo, I. & Kongprawechnon, W. (2008). Application of multiple tabu search algorithm to solve dynamic economic dispatch considering generator constraints: *Energy Conversion and Management* , Vol. 49, pp. 506–516.

Ramirez-Marquez, J. & David Coit, D. (2005). A Monte Carlo Simulation Approach for Approximating Multistate Two-Terminal Reliability: *Reliability Engineering & System Safety,* Vol. 87, No. 2.

Shinmori, S.; Koide, T. & Ishii, H. (1995). On lower bound for network reliability by edge-packing: *Transactions of the Japan Society for Industrial and Applied Mathematics,* Vol. 5, No. 2, pp. 139-151.

14

Hybrid Approaches Tabu Learning Algorithm Based Neural Network

Junfei Qiao, Jian Ye and Honggui Han
Beijing University of Technology
China

1. Introduction

Tabu search is a global search algorithm which is popular in recent years [F. Glover, 1989, 1990, 1997]. The main principle of tabu search is that it has some memory of the states that has already been investigated and it does not revisit those states. It considers the set of all possible neighbor states and takes the best one, but it will also take the best move in the neighborhood which might be worse than the current move. The tabu search focuses greatly on expanding the global search area and avoiding the search of the same area. It can always get much better global solutions. The tabu search uses a tabu list to memorize the visited states and keep from recurrent search. Aspiration criterion is set to activate the "tabued state" in the tabu list around which some good global states may be found [D. Cvijovic, 1995].

In the past decade, there has been a growing interest in applying neural network to many areas of science and engineering, such as pattern identification and image management [Jianming Lu, 2007], control[Jian-Xin Xu, 2007] and optimize[Z. S. H. Chan, 2005], communication [Kung S Y, et al. 1998] and so on. Basically, neural network is the computing system characterized by the ability to learn from examples rather than having to be programmed in a conventional way as used in control engineering [K.J. Astrom, B, 1989]. The broad use of neural network in many areas derived from its ability of approximating nonlinear functions. In theory, it has been proved that a three-layered neural network can approximate unknown functions to any degree of desired accuracy [K. Funahashi.1989; and K. Hornik, 1989].

This chapter is focused on the tabu learning algorithm based on neural network in which an unknown function is approximated. The input of the network is given by the values of the function variables and the output is the estimation of the function. In mathematical terms, the objective is to find appropriate values for the weights of the net which approximate the function best.

Gradient-based algorithms, especially the back propagation (BP) algorithm [L.M. Salchenberger, 1992] [P. Werbos, 1993] and its revised version [R. Parisi, 1996; and G. Zhou, 1998], are well known as a type of supervised learning for multilayered neural networks. The method of gradient descent is that a maximal "downhill" movement will eventually reach the minimum of the function surface over its parameter space by moving to the direction of the negative gradient.

However, this method has several inevitable drawbacks. First, it is prone to getting trapped in local minima of the error surface. Second, the training performance is sensitive to the choice of the learning rate and initial values of weights. For these reasons, many researches have been done to overcome these drawbacks especially the local convergence [Y. Izui, 1990 ; and S. Ma,1998].

Recently, some researches were done in the tabu learning algorithm [Junfei Qiao, et al. 2006]. In this chapter, we utilize the tabu search in the NN learning process to help the gradient technique "jump out of" the local minima and get a great improvement on the gradient technique. In order to test the ability of the tabu search to improve the NN learning, we introduce the most basic NN learning algorithm, BP algorithm, as the NN learning algorithm. The tabu search can also be combined with other improved learning algorithm.

The remainder of this chapter is organized as follows. Section 2 first describes the typical artificial neural network including the architecture and the learning algorithm. Then the target function used in this paper approximating the nonlinear function is introduced. In section 3, the tabu-based neural network learning algorithm, TBBP, is described. Section 4 illustrates the experiment and the result. In this section, both of the TBBP and BP are tested to approximate 6 different nonlinear functions. Eventually, section 5 gives the conclusion.

2. Neural network and target function

2.1 Neural network

We use the most employed architecture of NN in this section: a multilayer feed-forward network with a single hidden layer. The schematic representation of the neural network is showed in Fig. 1.

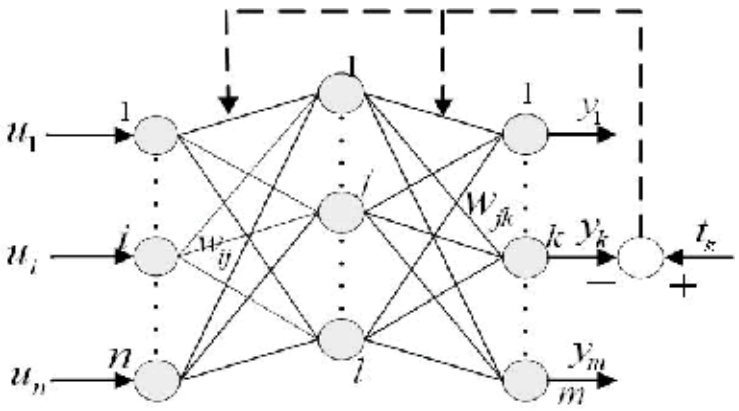


Fig. 1. The structure of the three-layered feed forward ANN

So the NN is presented as (N,W), where N is the set of nodes and W is the set of weights. There are 3 parts in N : N_I , input nodes; N_H , hidden nodes; N_O ,output nodes. If there are n variables in the nonlinear function that we want to approximate, $N_I = n$ .The neural network has l hidden neurons with a bias term in each hidden neuron. In the output layer, there are m nodes. To approximate the nonlinear function, there only has one single node in

the output layer as the approximated value of the function. In Fig.1, w_{ij} connects the i-th node in the input layer and the j-th node in the hidden layer. w_{jk} connects the j-th node in the hidden layer and the k-th node in the output layer.
Each node i in the input layer has a signal u_i as the input of the hidden layer. Each node j in the hidden layer receives a signal $In(j)$ according to

$$In(j) = \theta_j + \sum_{i=1}^{n} u_i \omega_{ij} \tag{1}$$

The most popular transform function in the hidden layer is the standard sigmoid function $f(x) = e^x/(e^x + 1)$, which is different from the linear function in both the input layer and the output layer. The output of the hidden node is $f(In(j))$. The output of the NN is

$$y_k = \theta_k + \sum_{j=1}^{m} \omega_{jk} f(In(j)) \tag{2}$$

Where θ_l and θ_m are the bias terms of the hidden layer and the output layer.

2.2 Target function

The process of determining the values of the weight parameters of the NN on the basis of a given data set is called supervised learning or training. The data set should contain a representative and sufficient number of input-target relations to allow an appropriate representation of the complex functional mapping of the problem. In the process of training the net, the purpose is to search the values of the weights that minimize the error across the training set. Once the optimization has been performed and the weights have been got, the NN is ready to approximate the function. The error function used in the optimization problem is called the target function.
The target function used in this paper is sum square error (sse) $E(w)$, which is:

$$E(w) = \frac{1}{2} \sum_{q=1}^{m} (T_q - Y_q)^2 \tag{3}$$

Where $Y_q = [y_1, y_2 \cdots y_k \cdots y_m]$ is the q-th output of the NN and $T_q = [t_1, t_2 \cdots t_k \cdots t_m]$ is the q-th target value. In Fig.1, $u_q = [u_1, u_2 \cdots u_i \cdots u_n]$ is the q-th input training data of the training data $\{\{u_1, T_1\}, \{u_2, T_2\} \cdots \{u_q, T_q\} \cdots \{u_n, T_n\}\}$. M is the number of training data.
The gradient descent method searches for the global optimum of the network weights. Each iteration t consists of two steps. First, partial derivatives $\partial E/\partial w$ are computed for each weight in the net. This can be exactly done by starting the computation at the output node k, and then successively computing the derivatives for the weights from w_{jk} to w_{ij} (the gradient information is successively moved from the output layer back towards the input layer). In the second step, weights are modified according to the expression:

$$w(t+1) = w(t) - \alpha \partial E(t) / \partial w(t) \tag{4}$$

Where the α is the learning rate. The error is computed for the new weights, t is increased in one unit and a new iteration begins.

In (3), W is the weight vector of the NN including the weights from the input to the hidden layer, the weights from the hidden to output layer, and the bias factor of the hidden and output nodes. W is represented as:

$$W = \left\{ \begin{bmatrix} \omega_{11} & \cdots & \omega_{1l} \\ \vdots & \ddots & \vdots \\ \omega_{n1} & \cdots & \omega_{nl} \end{bmatrix} \right\}, \begin{bmatrix} \omega_{11} & \cdots & \omega_{1m} \\ \vdots & \ddots & \vdots \\ \omega_{l1} & \cdots & \omega_{lm} \end{bmatrix}, (\theta_1, \theta_2 \cdots \theta_l), (\theta_1, \theta_2 \cdots \theta_m) \tag{5}$$

$E(w)$ can be treated as a nonlinear function of W for the same training data. Because the transform functions in the neural network is continuous, $E(w)$ is in a continuous multidimensional space of W. There are many concaves in this continuous space. The learning algorithm is to find the states at the bottom of the concaves which are the local optimal solutions of the function $E(w)$.

The traditional learning algorithms such as BP which is based on the gradient technique plunge into the local optimization inevitably.

3. The tabu based neural network learning algorithm

3.1 The tabu search in the NN learning

The tabu search (TS) which can be regarded as an iterative descent method is a global search algorithm. In the tabu search, the tabu list and the aspiration criterion are the most important factors for the tabu search to implement the memory and keep from the recurrent search. Tabu list (TL) is used to keep from recurrent search and the aspiration criterion (AC) is used to restart the search in the area in which there may be some superior solutions which can be found in the former chapters.

The simple Tabu Search method is mainly based on a random search. In the neural network learning, we use the weight vector W to compare to be the tabu object and mainly research about the weights. Neighbourhoods are randomly drawn points from uniform distribution. The neighborhood is a restricted region for each weight in the current weight. Hundreds of weights are randomly generated in this region and the best one is chosen. These weights are examined. The best solution in the neighborhood replaces the initial ones and the process is repeated. The weights in the tabu list represent the states which have been searched. The tabu list is generated by adding the last solution to the beginning of the list and discarding the oldest solution from it. If the new generated weight is not in the Tabu list (TL), the search goes on and this weight is added to the Tabu list (TL). To be rejected, all the weights of a new solution would need to be within a tabu area for any of the solutions in the tabu list (TL). The AC is used to activate the states that are tabued but around which there are some superior solutions. If there is one weight in the TL but it satisfies the AC, the complete test is also applied to this "tabu" weight.

3.2 The tabu based neural network Learning algorithm (TBBP)

The TBBP divides the learning algorithm into two steps, the superficial search (SS) and the deep search (DS). The superficial search mainly finds weights which seem to be big probability of being the "good global solution" in large numbers of neighbour solutions. The deep search trains these "good global solutions" to the end and gets the finial weights used to approximate the function.

The location of the original state has a great effect on the performance of algorithm. The original weight W_0 is randomly generated. If W_0 is trained directly with BP from the original state to the end, it can only get a local optimal solution which is greatly based on the original state W_0. From these different W_0, only some random local optimal solutions are reached. The SS is implemented to filtrate the initial solutions and pick out some good solutions for the deep search. In addition, there is such a kind of weight $W_{01}, W_{02} \cdots W_{0n}$ which are in the same concave of the $E(W)$ space. If the weights $W_{01}, W_{02} \cdots W_{0n}$ are directly searched to the end, the same concave are searched n times and only the same local optimal solution is got. Lots of time is wasted during the recurrent search. The TBBP uses the SS to keep from recurrent search in the same concave.

The superficial search trains the original weight W_0 to a state $W_0^{'}$ which has a depth but is not at the bottom of the concave. $W_0^{'}$ is defined to be the superficial state. TBBP put these superficial states into the tabu list as the tabu object. If $W_0^{'}$ is in the tabu list, it presents that $W_0^{'}$ is in a searched concave and TBBP don't go on to search deeply, otherwise $W_0^{'}$ is considered to be in a new concave and should be deeply searched.

Whether the concave has been searched or not is very important to avoid the recurrent search. The superficial state $W_0^{'}$ which has been deeply searched is in the TL. If a new generated superficial state ($W_i^{'}$) is in the tabu area (TA) of a tabu object ($W_{t(j)}^{'}$) in the TL, TBBP consider $W_i^{'}$ to be in a concave which has been deeply searched with a biggish probability. Then $W_i^{'}$ is given up and the next superficial state will be tested.

There may be some $W_i^{'}$ which are in the TL (corresponding to $W_{t(j)}^{'}$) but satisfy

$$E(W_i^{'}) < (1 - AC) \cdot E(W_{t(j)}^{'}) \; or E(W_i^{'}) > (1 - AC) \cdot E(W_{t(j)}^{'}) \tag{6}$$

where $E(W_i^1)$ is the sum of error evaluated at the superficial state $W_i^{'}$.

TBBP uses (6) as the aspiration criterion (AC) to activate $W_i^{'}$ which is tabued and continue to train it deeply. If the "tabued" superficial state, $W_i^{'}$, in the tabu area of $W_{t(j)}^{'}$ but fulfils (6), TBBP consider that $W_i^{'}$ jump out from the concave in which $W_{t(j)}^{'}$ is with a biggish probability.

When the SS have picked out the proper weights, the TBBP trains the NN from these weights. This is called the deep search. The DS trains the NN by using the back propagation algorithm.

3.3 The important parameters in TBBP

3.3.1 The tabu area

The probability of finding weights that they are exactly identical is zero because the weights are real values. TBBP uses tabu area (TA) to relax the strict comparison between the new weight and the weights in the tabu list.

The value of TA is very important to keep from recurrent search. If TA is too big, the algorithm may tabu some superior states which are in the concave that don't have been search before and the search area is decreased. Some good global solutions maybe lost. If TA is set to be too small, some solutions in the same concave are searched deeply and the recurrent search TBBP can't be avoided effectively.

In the experiment, many values were tested. In this paper, TA is set in two different values, 0.03 and 0.01, to illustrate how to choose the TA. 100 neighbor solutions $W_{0j}^{'}$ $(j = 1,2\cdots 100)$ are generated in the neighborhood of the same superficial weight $W_0^{'}$. The size of the neighborhood is set to be the TA. TBBP deeply search the 100 neighbor solutions and get the corresponding deep state $W_{0j}^{''}$. The $E(W_{0j}^{''})$ sse_{0j} are compared to $E(W_0^{''})$ sse_0.

Fig.2 shows the different values of D after training 10000 epochs and 100000 epochs when $TA = 0.03$ and $TA = 0.01$.

$$D = (sse_{0j} - sse_0) \big/ sse_0 \tag{7}$$

In Fig.2, whether the TA is set to be 0.03 or 0.01, all the D are within ± 0.03 after training 100000 epochs. Some deviations are big when set TA to be 0.03 after training 10000 epochs. If $TA = 0.01$, many states are considered to be in the different concaves but actually they are in the same concave. Therefore, lots of time is wasted and TBBP can't avoid recurrent search effectively.

According to this experiment, it is obvious that $TA = 0.03$ is better. To be tabued, all the values in the new weight vector should be within this range for any of the tabu objects in the TL.

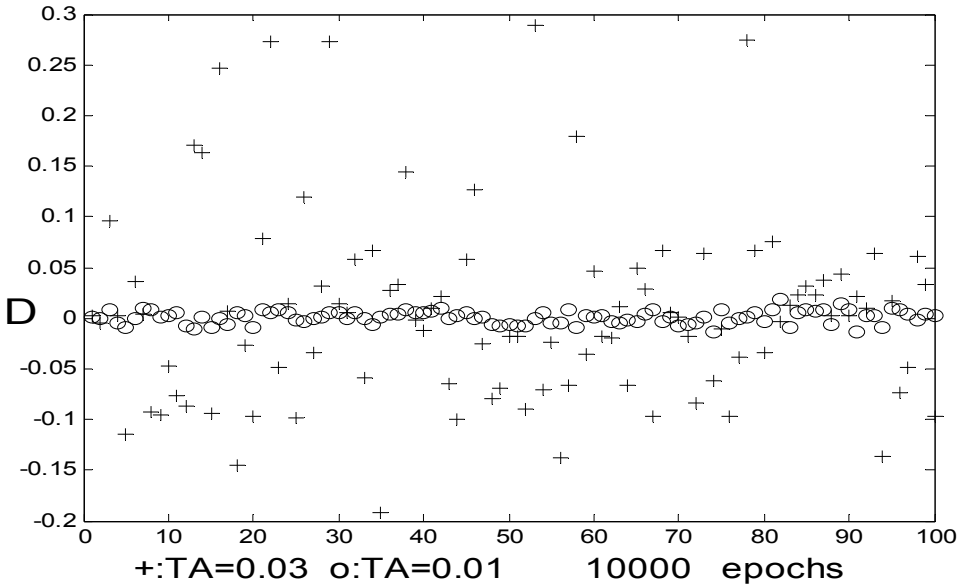
3.3.2 The depth of the superficially search (DSS)

The DSS is the most important parameter in TBBP. If the SS doesn't taken place or the DSS is not deep enough, recurrent search can't be avoided in the global area. If the superficial search is too deep, the neighbor solutions can hardly jump out of the original concave and the search area isn't extended adequately.

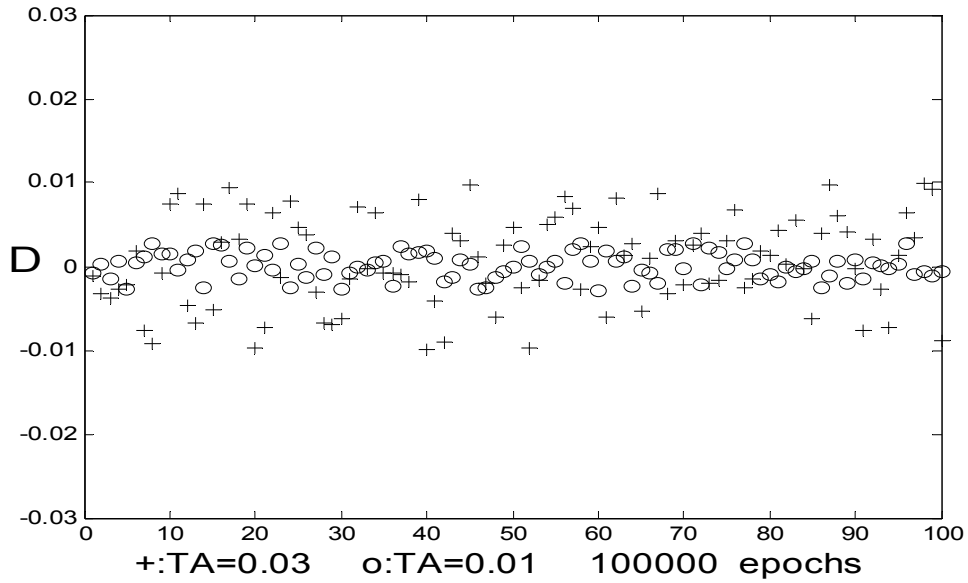
BP algorithm can be simply described as in (4). We set

$$g_k = \partial E(t) / \partial w(t) \tag{8}$$

where g_k is the current gradient.



(a)



(b)

Fig. 2. The different deviations of sse_{0j} from sse_0 .

During the training, g_k may increase in some periods, but there is the trend that g_k decrease when the search depth is increased during the whole training process. Therefore,

TBBP cease the superficial search according to the gradient g. When $g_k = g$, the superficial search stops and the current weight of the NN is set to be the superficial state.

The deep search (DS) is carried out with the 100 neighbor solutions ($W_{ij}^{'}$) which are generated in the neighborhood of the superficial states $W_i^{'}$. Train $W_{ij}^{'}$ deeply and get the deep states $W_{ij}^{''}$. $E(W_{ij}^{''})$ (sse_{ij}) are compared to $E(W_i^{''})$ (sse_i) where $W_i^{''}$ is deeply trained from the original superficial state $W_i^{'}$.

The Fig.3 and Fig.4 show the different deviations of sse_{ij} from sse_i.

$$D = (sse_{ij} - sse_i) / sse_i \tag{9}$$

In the experiment, g is set in three different levers, 0.01, 0.05 and 0.1. If DSS is set to be to small ($g < 0.01$), the algorithm can't jump out of the local minima effectively and some superior solutions may be lost. If DSS is set to be too big ($g > 0.1$), the recurrent search can't be avoided and lots of time is wasted. The deviation is the greatest when $g = 0.05$.

TBBP set DSS to be $g = 0.05$. It considers that the search area are extended most widely from the original superficial state when DSS=0.05. Meanwhile, the recurrent search can be kept from most effectively when DSS=0.05.

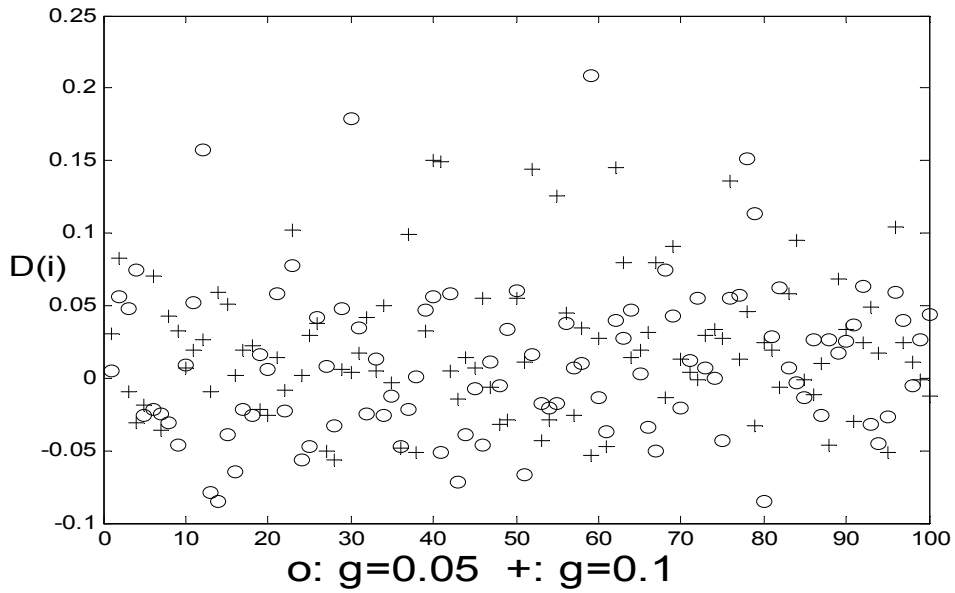


Fig. 3. The different deviations of sse_{0j} from sse_0 when g is set to be too big

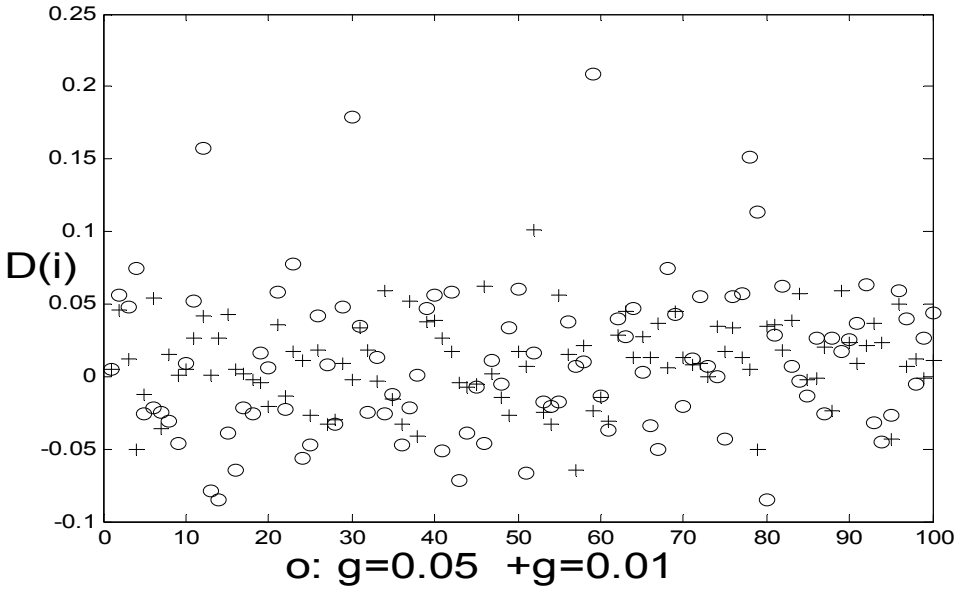


Fig.4. The different deviations of sse_{0j} from sse_0 when g is set to be too small

3.4 The Basic Step of TBBP

In this section, we give the detailed description of TBBP. It mainly consists of seven steps as follows:

(1) Initialization.

This step generates an original weight vector W_i which is described in (5) randomly.

(2) Superficial Search.

i. The initial weight W_i is trained with BP algorithm.

ii. The superficial state $W_i^{'}$ and $E(W_i^{'})$ are got.

iii. The $W_i^{'}$ is tested to decide whether the deep search is implemented or not. If $W_i^{'}$ is tabued and doesn't satisfy AC, go to Step (1) to generate the next initial weight; otherwise, go to Step (3) and add it into the Tabu List.

(3) Deeply search the superficial state $W_i^{'}$.

i. Deeply search the $W_i^{'}$ and get the corresponding deep state $W_i^{''}$.

ii. Evaluate $E(W_i^{''})$ (sse_i) and set it to be the best weight in this neighborhood. $W_{best(i)} = W_i^{''}$.

(4) Generate a neighbor solution $W_{ij}^{'}$ around the superficial state $W_i^{'}$ randomly and evaluate $E(W_{ij}^{'})$.

(5) Test the superficial state $W_{ij}^{'}$ in the neighbourhood of $W_i^{'}$.

If $W_i^{'}$ is in the TL and doesn't satisfy the AC, go to Step (4) to generate the next $W_{ij}^{'}$; otherwise, go to Step (6) to search $W_{ij}^{'}$ deeply.

(6) Deeply search $W_{ij}^{'}$.

i. Train $W_{ij}^{'}$ with BP algorithm and get $W_{ij}^{''}$.

ii. If $E(W_{ij}^{''}) < E(W_{best(i)})$, set $W_{best(i)} = W_{ij}^{''}$.

iii. If $W_{ij}^{'}$ have been generated, go to Step (1) to get another original weight W_i; otherwise, go to Step (4) to generate the next $W_{ij}^{'}$.

iv. Go to Step (7) if the maximal num of the neighborhood is reached.

(7) Get the weight generated by TBBP.

Compare all the $E(W_{best(i)})$ and get the best one as the solution of TBBP.

4. Experimental evaluation

In this section, we first introduce our experiments and then give the results.

4.1 Experiments

The TBBP and BP algorithm both are written in C and all the programs are complied in VC++ 6.0. All the figures (including the Figures in Section 3) are drawn by Matlab 6.0 using the data generated by the program.

The important parameters of TBBP such as the DSS and TA are discussed in Section 3. In this subsection, the experiments are performed using the value selected in Section 3.

In order to test the ability of TBBP to approximate the nonlinear function, we choose 6 nonlinear objective functions to test. They are

$$y = x_1 + x_2 \tag{a}$$

$$y = x_1 x_2 \tag{b}$$

$$y = x_1 / |x_2| + 1 \tag{c}$$

$$y = x_1^2 - x_2^3 \tag{d}$$

$$y = x_1^3 - x_1^2 \tag{e}$$

$$y = \alpha_1 \arctan(x_1 - \beta_1) + \alpha_2 \arctan(x_2 - \beta_2) + \alpha_3 \arctan(x_3 - \beta_3) + \lambda \tag{f}$$

Where $\lambda = -\alpha_1 \arctan(-\beta_1) - \alpha_2 \arctan(-\beta_2) - \alpha_3 \arctan(-\beta_3)$.

Because TBBP is designed to test the superior effect of tabu search used in NN learning, a simple 3-layered feed forward neural network which is described in Fig.1 is chosen for all

the 6 nonlinear functions. There are six nodes in the hidden layer. All the nets have only one output node whose output is severed as the approximating function value.

The training data of function (a)(b)(c)(d)(e) are all generated randomly from $x_i \in [-1,1]$. The training data of function (f) is generated from $x_i \in [0,200]$.

In order to test the TBBP's ability of forecasting the nonlinear function in the unknown areas, the interpolation and extrapolation data are set to be tested. The interpolation data are generated randomly in the same range of the training data. The extrapolation data are generated outside the range of the training data. It is $x_i \in [-2,-1]$ or $x_i \in [1,2]$ for function (a)(b)(c)(d)(e), and $x_i \in [200,400]$ for function (f).

In TBBP, 200 superficial states are tested for each function and 100 neighbor solutions are generated in the neighborhood of each superficial state. In BP, We set the momentum to be 0.9 and train the neural network 50000 iterations.

In order to show how greatly the tabu search works to avoid recurrent search, counters are set to record the times the tabu take place.

4.2 Experimental results

For all the 6 nonlinear functions, the best sum square error (sse) generated in both TBBP and BP are shown in Table1.

		y_1	y_2	y_3	y_4	y_5	y_6
Training samples	TBBP	7.41E-05	7.65E-04	4.29E-02	2.82E-03	8.91E-04	5.41E+01
	BP	2.18E-02	2.46E-02	1.12E-01	1.07E-01	1.78E-01	4.39E+02
A	TBBP	3.42E-03	9.82E-03	1.90E-01	3.58E-01	8.12E-02	6.28E+02
	BP	9.48E-01	3.28E-01	4.57E+00	2.07E+00	3.77E+00	9.27E+03
B	TBBP	6.71E-01	5.34E-01	4.21E+01	1.85E+01	4.33E+01	3.42E+04
	BP	3.75E+01	6.71E+01	1.92E+02	4.09E+01	1.70E+02	9.85E+05

Table 1. The comparison of sum square error (sse) between the two algorithms

Results in Table 1 present that the sse generated by TBBP are obvious smaller than the BP's. It illustrates that the TBBP can approximate the function in all the 3 data including training data, interpolation data and extrapolation data.

The advantage of the approximation in the extrapolation data of TBBP is smaller than the corresponding one in the interpolation data. This is because the interpolation data are generated in the same range of training data. But the extrapolation data is outside the range of the training data.

The sse of TBBP is not very small, Especially in extrapolation data for function (c) (d) (e) (f). Why is the sse of the TBBP not smaller enough (maybe bigger than the sse of GA)? TBBP also uses the BP algorithm as the learning strategy. Actually, we only use tabu search in BP

to test its ability in NN learning. We can also corporate tabu search with other methods such as GA and compare the result with the GA algorithm.

Table 2 shows the times the tabu take place in every 100 neighbor searches of each superficial state. In the table, max times, min times and the mean times the tabu take place are given. From Table 2, we can conclude that the tabu search have played an important role in the recurrent search. Because neighbor solutions are generated randomly, there is no rule of the data in the Table 2.

	(a)	(b)	(c)	(d)	(e)	(f)
min	15	18	23	7	20	14
max	41	43	39	28	32	44
mean	29	31	32	17	26	22

Table 2. Times the tabu take place in every 100 neighbor solutions of each superficial state

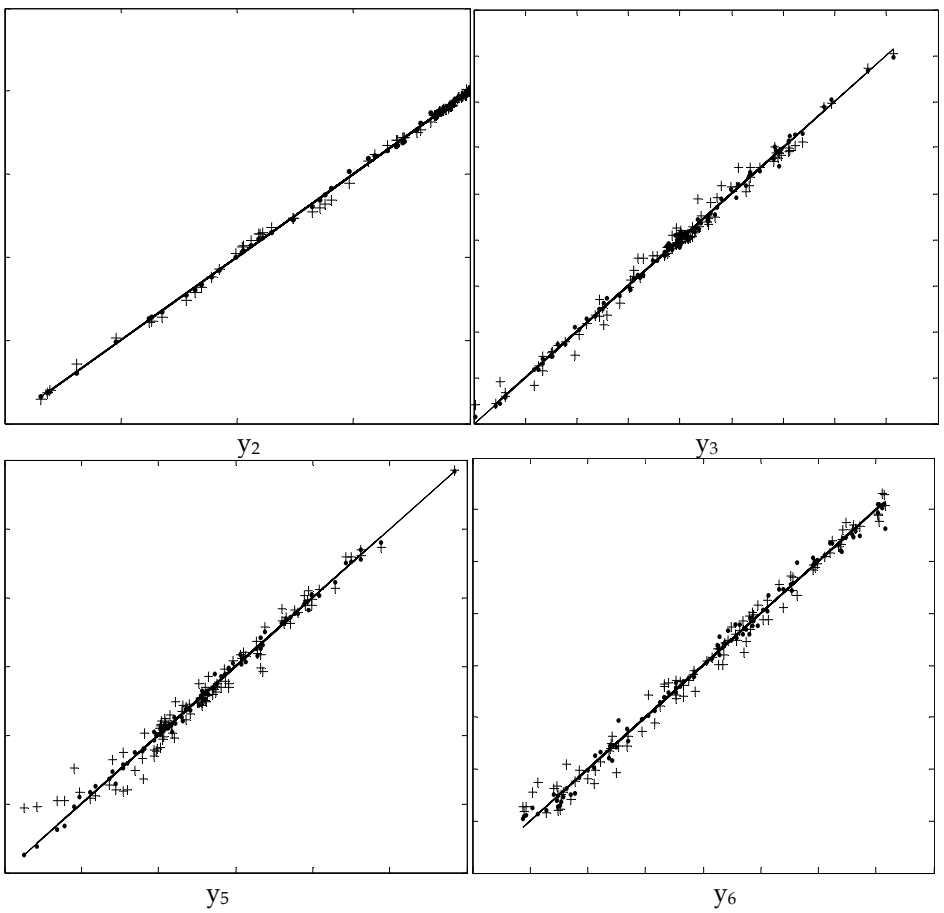


Fig. 5. The comparison of two algorithms in forecasting the interpolation data for different functions. '+' represent BP and '.' represent TBBP

Since the ability of forecasting the function in the non-training data is the most important thing, Fig. 5 is provided to illustrate the different values forecasted by TBBP and BP in the interpolation data for function (b)(c)(e)(f). X axis represents the true value for the interpolation data, Y axis represents the value forecasted by the NN. The line denotes the destination values.

$E(W)$ is smaller when the point is closer to the line. In Fig.5, the point forecasted by TBBP is much closer than those forecasted by BP. Consequently the TBBP can approximate the nonlinear function much more accurately.

5. Conclusions

In this paper, we propose an approach, TBBP, to solve the local convergence of the gradient method. The tabu search can jump out of the local minima, expand the search area and avoid the recurrent search. By using the tabu search in the NN learning, TBBP shows a great improvement of jumping out from the local minimal. It can also approximate the nonlinear function in unknown area. Experiment results show that:

1) TBBP can jump out of the local minimal to extend the search in the global space;

2) Recurrent search can be kept from effectively and lots of time is saved.

This chapter shows the efficiency of integrating the tabu search into the neural network learning. It also gives us a clue that the tabu search is feasible to cooperate with other NN learning algorithms (not only BP).

6. Acknowledgements

This work was supported by the National Science Foundation of China under Grant 60674066; and by the Beijing Education Committee Technology Development Foundation under Grant KM20061005019; and the Beijing Elitists Foundation under Grant 20061D0501500203; and the Beijing Municipality 'New Star' program.

7. References

F. Glover. (1989). Tabu search—Part I, ORSA J. Comput, Vol 1. NO.3, 190–206.

F. Glover. (1990). Tabu search—Part II, ORSA J. Comput, Vol 2. NO.1, 4–32.

F. Glover, M. Laguna.(1997). Tabu search, Boston: Kluwer.

D. Cvijovic, J. Klinowski. (1995). Taboo search: an approach to the multiple minima problem, Science Vol 267. NO.3, 664–666.

Jianming Lu, Xue Yuan, and Takashi Yahagi.(2007). A Method of Face Recognition Based on Fuzzy c-Means Clustering and Associated Sub-NNs. IEEE Trans on Neural Network, vol. 18, NO. 1, 150-151.

Jian-Xin Xu, Ying Tan.(2007). Nonlinear Adaptive Wavelet Control Using Constructive Wavelet Networks. IEEE Trans on Neural Network, vol. 18, NO. 1, 115-128.

Z. S. H. Chan and N. Kasabov.(2005). Fast neural network ensemble learning via negative-correlation data correction. IEEE Trans. Neural Network., Vol. 16, NO. 6, 1707–1710.

Kung S Y, et al.(1998). Neural Networks for Intelligent Multimedia Processing. Proc IEEE, 1244-1272.

K.J. Astrom, B. Wittenmark.(1989). Adaptive Control. Reading, MA: Addison-Wesley.

K. Funahashi.(1989).On Approximate Realization of Continuous Mappings by Neural Network, IEEE Trans. Neural Netwoks. Vol2. NO.3, 183-192.

K. Hornik, M. Stinchcombe, H. White. (1989). Multilayer feed-forward network are universal approximators, IEEE Trans. Neural Networks.Vol2. NO.5, 359-366.

L.M. Salchenberger, E.M. Cinar, N.A. Lash. (1992). Neural networks: a new tool for predicting thrift failures, Decision Sciences 23,899-916.

P. Werbos.(1993).The roots of backpropagation from ordered derivatives to neural networks and political forecasting, John Wiley & Sons Inc, New York.

R. Parisi, E.D.Di Claudio. (1996). A generalized learning paradigm exploiting the structure of feedforward neural networks, IEEE Trans. Nueral Networks Vol7.NO.6, 1450-1459.

G. Zhou, J. Si. (1998).Advanced neural network training algorithm with reduced complexity based on Jacobian deficiency, IEEE Trans. Neural Networks .Vol9.NO.3, 448-453.

Y. Izui, A. Pentland. (1990).Analysis of neural networks with redundancy, Neural Computation Vol2, 226-238.

S. Ma, C. Ji..(1998).A unified approach on fast training of feedforward and recurrent networks using EM algorithm, IEEE Trans. Signal Processing, Vol 46,2270-2274.

Jian Ye, Junfei Qiao, Xiaogang Ruan.(2006) A Tabu Based Neural Network Learning Algorithm. Neurocomputing, Vol 10.

Edited by Wassim Jaziri

The goal of this book is to report original researches on algorithms and applications of Tabu Search to real-world problems as well as recent improvements and extensions on its concepts and algorithms. The book,Äô Chapters identify useful new implementations and ways to integrate and apply the principles of Tabu Search, to hybrid it with others optimization methods, to prove new theoretical results, and to describe the successful application of optimization methods to real world problems. Chapters were selected after a careful review process by reviewers, based on the originality, relevance and their contribution to local search techniques and more precisely to Tabu Search.

ISBN 978-953-51-5831-8

Photo by klikk / iStock

IntechOpen