IntechOpen

Vehicle Routing Problem

Edited by Tonci Caric and Hrvoje Gold

Vehicle Routing Problem

Edited by

Tonci Caric and Hrvoje Gold

Vehicle Routing Problem
http://dx.doi.org/10.5772/64
Edited by Tonci Caric and Hrvoje Gold

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

Vehicle Routing Problem
Edited by Tonci Caric and Hrvoje Gold

p. cm.

ISBN 978-953-7619-09-1

eBook (PDF) ISBN 978-953-51-6401-2

Preface

The Vehicle Routing Problem (VRP) dates back to the end of the fifties of the last century when Dantzig and Ramser set the mathematical programming formulation and algorithmic approach to solve the problem of delivering gasoline to service stations. Since then the interest in VRP evolved from a small group of mathematicians to the broad range of researchers and practitioners, from different disciplines, involved in this field today.

The VRP definition states that m vehicles initially located at a depot are to deliver discrete quantities of goods to n customers. Determining the optimal route used by a group of vehicles when serving a group of users represents a VRP problem. The objective is to minimize the overall transportation cost. The solution of the classical VRP problem is a set of routes which all begin and end in the depot, and which satisfies the constraint that all the customers are served only once. The transportation cost can be improved by reducing the total travelled distance and by reducing the number of the required vehicles.

The majority of the real world problems are often much more complex than the classical VRP. Therefore in practice, the classical VRP problem is augmented by constraints, such as vehicle capacity or time interval in which each customer has to be served, revealing the Capacitated Vehicle Routing Problem (CVRP) and the Vehicle Routing Problem with Time Windows (VRPTW), respectively. In the last fifty years many real-world problems have required extended formulation that resulted in the multiple depot VRP, periodic VRP, split delivery VRP, stochastic VRP, VRP with backhauls, VRP with pickup and delivering and many others.

VRP is NP hard combinatorial optimization problem that can be exactly solved only for small instances of the problem. Although the heuristic approach does not guarantee optimality, it yields best results in practice. In the last twenty years the meta-heuristics has emerged as the most promising direction of research for the VRP family of problems.

This book consists of nine chapters. In the first four chapters the authors use innovative combinations of methods to solve the classically formulated VRP. The next two chapters study the VRP model as a tool for formalizing and solving problems not often found in the literature. The solving procedures in the next two chapters use the evolutionary multi-criteria optimization approach. The last chapter is focused on multi-resource scheduling problem. A brief outline of chapters is as follows:

Chapter 1 "Scatter Search for Vehicle Routing Problem with Time Windows and Split Deliveries" considers the scatter search as a framework which provides a means to combine solutions, diversify, and intensify the meta-heuristic search process. The initial solution is an extension of Solomon's sequential insertion heuristic I1. The experiments are carried out on Solomon's problems with augmented demands which allow more than one delivery per customer.

Chapter 2 "A Modelling and Optimization Framework for Real-World Vehicle Routing Problems" presents an integrated modelling and optimization framework for solving

complex and practical VRP. The modular structure of the framework, a script based modelling language, a library of VRP related algorithms and a graphical user interface give the user both reusable components and high flexibility for rapid prototyping of complex VRP. The algorithm and the performance measure protocol is explained and implemented on the standard benchmark and on practical VRPTW problems.

Chapter 3 "An Effective Search Framework Combining Meta-Heuristics to Solve the Vehicle Routing Problems with Time Windows" mainly considers combining of the two well-known methods (guided local search and tabu search) for solving VRPTW where the choice of the search method in each iteration is controlled by simulated annealing. The initial solution is obtained by push-forward insertion heuristics and the virtual vehicle heuristics. The approach is verified on the standard Solomon's benchmark.

Chapter 4 "A Hybrid Ant Colony System Approach for the Capacitated Vehicle Routing Problem and the Capacitated Vehicle Routing Problem with Time Windows" deals with hybrid ant colony with local search as the mechanism to solve CVRP and CVRPTW problems. 2-opt and saving heuristics participate in hybridization. To improve the solution the algorithm introduces the simulated annealing in the phase of the pheromones updating. Hybrid Ant Colony System is tested on the classical CVRP and VRPTW problems.

Chapter 5 "Dynamic Vehicle Routing for Relief Logistics in Natural Disasters" examines the dynamic vehicle routing problem for relief logistics (DVRP-RL) in natural disasters as a support to the relief management. Two-phase heuristic, route construction and route improvement for DVRP-RL is proposed. The solution procedure resolves the current change of constraints generated by new events in the field in such a way which updates information in DVRP-RL and initiates route construction or route improvement. Differences between features of VRPTW and DVRP-RL are presented. Routing of vehicles in constrained environment of emergency is computed and analysed.

Chapter 6 "Cumulative Vehicle Routing Problems" develops formulation of cumulative VRP, named CumVRP, which is based on capacitated VRP extended by the cost function defined as a product of the distance travelled and the flow on that arc. m-Travelling Repairman Problem, Energy Minimizing Vehicle Routing Problem and Average Distance-Minimizing School-bus Routing Problem can be formulated as the special cases of CumVRP. Numerical examples of CumVRP formulation focusing on the collection case of the Energy Minimizing VRP are provided.

Chapter 7 "Enhancing Solution Similarity in Multi-Objective Vehicle Routing Problems with Different Demand Periods" describes the problem of multiple-objective VRP constituted of two periods with different demands. The objectives are minimization of the maximum routing time, minimization of the number of vehicles and maximization of the similarity of solutions. To obtain a similar set of solutions in the normal and high demand period two-fold evolutionary multi-criteria optimization algorithm is applied. The simulation result on a multi-objective VRP with two periods with different demands are presented.

In Chapter 8 "A Multiobjectivization Approach for Vehicle Routing Problems" the single-objective optimization CVRP problem is translated into multi-objective optimization problem using the concept of multiobjectivization. On the translated problem the evolutionary multi-criteria optimization algorithm is applied. Experimental results indicate that multiobjectivization using additional objectives is more effective than using either objective alone.

Chapter 9 "Resources Requirement and Routing in Courier Service" studies multi-resource scheduling in pickup and delivery operations occurring mostly in the courier service. The objective is to examine the possible cost savings and computational time required as delivery resources operate in some cooperative modes. Computational analysis based on the real-life and simulated data is carried out.

This book presents recent improvements, innovative ideas and concepts regarding the vehicle routing problem. It will be of interest to students, researchers and practitioners with knowledge of the main methods for the solution of the combinatorial optimization problems.

July 2008

Editor

Tonči Carić
Hrvoje Gold
Uniaversity of Zagreb
Faculty of Traffic and Transport Sciences
Vukelićeva 4, HR-10000 Zagreb,
Croatia

Contents

1

Scatter Search for Vehicle Routing Problem with Time Windows and Split Deliveries

Patrícia Belfiore[1], Hugo Tsugunobu[2] and Yoshida Yoshizaki[2]
[1]Department of Production Engineering – FEI University Center,
São Bernardo do Campo-SP,
[2] Department of Production Engineering – University of São Paulo (USP)
Brazil

1. Introduction

The classical vehicle routing problem (VRP) aims to find a set of routes at a minimal cost (finding the shortest path, minimizing the number of vehicles, etc) beginning and ending the route at the depot, so that the known demand of all nodes are fulfilled. Each node is visited only once, by only one vehicle, and each vehicle has a limited capacity. Some formulations also present constraints on the maximum traveling time.

The VRPSD is a variation of the classical VRP, where each customer can be served by more than one vehicle. Thus, for the VRPSD, besides the delivery routes, the amount to be delivered to each customer in each vehicle must also be determined. The option of splitting a demand makes it possible to service a customer whose demand exceeds the vehicle capacity. Splitting may also allow decreasing costs. The vehicle routing problem with time windows and split deliveries (VRPTWSD) is an extension of the VRPSD, adding to it the time window restraints.

Lenstra and Rinnooy Kan (1981) have analyzed the complexity of the vehicle routing problem and have concluded that practically all the vehicle routing problems are NP-hard (among them the classical vehicle routing problem), since they are not solved in polynomial time.

According to Solomon and Desrosiers (1988), the vehicle routing problem with time windows (VRPTW) is also NP-hard because it is an extension of the VRP.

Although the vehicle routing problem with split deliveries (VRPSD) is a relaxation of the VRP, it is still NP-hard (Dror and Trudeau, 1990, Archetti et al., 2005).

Therefore, the VRPTWSD is NP-hard, since it is a combination of the vehicle routing problem with time windows (VRPTW) and the vehicle routing problem with split delivery (VRPSD), and that makes a strong point for applying heuristics and metaheuristic in order to solve the problem.

This work develops a scatter search (SS) algorithm to solve a vehicle routing problem with time windows and split deliveries (VRPTWSD). To generate the initial solutions of SS we propose an adaptation of the sequential insertion heuristic of Solomon (1987).

Ho and Haugland (2004) modified the customers' demands of the Solomon's test problems in order to perform split deliveries. Numerical results of SS are reported as well as comparisons with the Ho and Haugland algorithm.

The sequence of the chapter is described next. Section 2 describes the literature review for VRPSD and its extensions. Section 3 presents the problem definition, including the mathematical formulation. Section 4 describes the scatter search overview. Section 5 describes the heuristic and the scatter search approach proposed in order to solve the model. Section 6 presents the computational results. Finally, some conclusions are drawn in the last section.

2. Literature review for the VRPSD (TW)

The vehicle routing problem with split deliveries (VRPSD) was introduced in the literature by Dror and Trudeau (1989, 1990), who presented the mathematical formulation of the problem and analyzed the economy that can be made when it is allowed that a customer is fulfilled by more than one vehicle, economy both related to number of vehicles and total distance traveled.

Dror et al. (1994) have presented an integer programming formulation of the VRPSD and have developed several families of valid inequalities, and a hierarchy between these is established. A constraint relaxation branch and bound algorithm for the problem was also described.

Frizzell and Giffin (1992, 1995) have developed construction and improvement heuristics for the VRPSD with grid network distances. In their second publication they also considered time windows constraints.

Mullaseril et al. (1997) have described a feed distribution problem encountered on a cattle ranch in Arizona. The problem is cast as a collection of capacitated rural postman problem with time windows and split deliveries. They presented an adaptation of the heuristics proposed by Dror and Trudeau (1990).

Belenguer et al. (2000) proposed a lower bound for the VRPSD based on a polyhedral study of the problem. This study includes new valid inequalities. The authors developed a cutting-plane algorithm to solve small instances. For bigger instances, integer values are obtained via branch-and-bound.

Archetti et al. (2006b) have done the worst-case performance analysis for the vehicle routing problem with split deliveries. The authors have shown that the cost savings that can be realized by allowing split deliveries is at most 50%. They also study the variant of the VRPSD in which the demand of a customer may be larger than the vehicle capacity, but where each customer has to be visited a minimum number of times. Archetti et al. (2006a) have described a tabu search algorithm for the VRPSD. At each iteration, a neighbour solution is obtained by removing a customer from a set of routes where it is currently visited, and by inserting it either into a new route, or into an existing route which has enough residual capacity. The algorithm also considers the possibility of inserting a customer into a route without removing it from another route.

Ho and Haugland (2004) have developed a tabu search algorithm for the VRPTWSD. The first stage constructs a VRP solution using node interchanges, and the second stage improves the VRP solution by introducing and eliminating splits. There is a pool of solutions that are defined by different move operators. The best solution in the current pool is always chosen.

Belfiore (2006) proposed a scatter search algorithm to solve a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries that occurs in a major

Brazilian retail group. The results show that the total distribution cost can be reduced significantly when such methods are used.
In this chapter, we developed a new scatter search algorithm to solve the VRPTWSD. The results will be compared with Ho and Haugland modified problems.

3. Problem formulation and definitions

In this section we define the problem under study, and the notation used throughout the chapter.
Customers
The problem is given by a set of customers $N = \{1, 2, .., n\}$, residing at n different locations. Every pair of locations (i, j), where $i, j \in N$ and $i \neq j$, is associated with a travel time t_{ij} and a distance traveled d_{ij} that are symmetrical ($t_{ij} = t_{ji}$ and $d_{ij} = d_{ji}$). Denote by q_i, $i = 1, 2, ..., n$, the demand at point i. The central depot is denoted by 0.
Fleet of vehicles
The customers are served from one depot with a homogeneous and limited fleet. The vehicles leave and return to the depot. There is a set V of vehicles, $V = \{1, ..., m\}$, with identical capacities. The capacity of each vehicle $k \in V$ is represented by a_k.
We let $R_i = \{r_i(1), ..., r_i(n_i)\}$ denote the route for vehicle i, where $r_i(j)$ is the index of the jth customer visited and n_i is the number of customers in the route. We assume that every route finishes at the depot, i.e. $r_i(n_i + 1) = 0$.
Time Windows
Each customer $i \in N$ has a time windows, i.e. an interval $[e_i, l_i]$, where $e_i \leq l_i$ which corresponds, respectively, to the earliest and latest time to start to service customer i. Let s_i be the service time at customer i.
Split deliveries
The demand of a customer may be fulfilled by more than one vehicle. This occurs in all cases where some demand exceeds the vehicle capacity, but can also turn out to be cost effective in other cases.
The decision variables of the model are:

$x_{ij}^k =$ 1, if j is supplied after i by vehicle k;
0, otherwise.

$b_i^k =$ moment at which service begins at customer i by vehicle k, $i = 1, .., n \quad k = 1, ..., m$

$y_i^k =$ fraction of customer's demand i delivered by vehicle k.

The objective of the model is to minimize the total distance traveled respecting the time window constraints. The mathematical programming formulation is presented below based on Dror and Trudeau (1990) and Ho and Haugland (2004).

The objective function can be written as follows:

$$\min \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{m} d_{ij} x_{ij}^{k}$$

The model constraints are:

$$\sum_{j=1}^{n} x_{0j}^{k} = 1 \qquad k = 1, \ldots, m$$

Constraint (1) guarantees that each vehicle will leave the depot and arrive at a determined customer.

$$\sum_{i=0}^{n} x_{ip}^{k} - \sum_{j=0}^{n} x_{pj}^{k} = 0 \qquad p = 0, \ldots, n; \qquad k = 1, \ldots, m$$

Constraint (2) is about entrance and exit flows, guarantees that each vehicle will leave a determined customer and arrive back to the depot.

$$\sum_{k=1}^{m} y_{i}^{k} = 1, \qquad i = 1, \ldots, n$$

Constraint (3) guarantees that the total demand of each customer will be fulfilled.

$$\sum_{i=1}^{n} q_{i} y_{i}^{k} \leq a_{k} \qquad k = 1, \ldots, m$$

Constraint (4) guarantees that the vehicle capacity will not be exceed.

$$y_{i}^{k} \leq \sum_{j=0}^{n} x_{ji}^{k} \qquad i = 1, \ldots, n; \quad k = 1, \ldots, m$$

Constraint (5) guarantees that the demand of each customer will only be fulfilled if a determined vehicle goes by that place. We can notice that, adding to constraint (5) the sum of all vehicles and combining to equation (3) we have the constraint $\sum_{k=1}^{m} \sum_{i=0}^{n} x_{ij}^{k} \geq 1 \quad j = 0, \ldots, n$, which guarantees that each vertex will be visited at least once by at least one vehicle.

$$b_{i}^{k} + s_{i} + t_{ij} - M_{ij}(1 - x_{ij}^{k}) \leq b_{j}^{k} \qquad i = 1, \ldots, n; \; j = 1, \ldots, n; \; k = 1, \ldots, m$$

Equation (6) sets a minimum time for beginning the service of customer *j* in a determined route and also guarantees that there will be no sub tours. The constant M_{ij} is a large enough number, for instance, $M_{ij} = l_{i} + t_{ij} - e_{j}$.

$$e_i \le b_i^k \le l_i \quad i = 1,...,n$$

Constraint (7) guarantees that all customers will be served within their time windows.

$$y_i^k \ge 0 \quad i = 1,...,n; \quad k = 1,...,m$$
$$b_i^k \ge 0 \quad i = 1,...,n; \quad k = 1,...,m$$

Equation (8) guarantees that the decision variables y_i^k and b_i^k are positive.

$$x_{ij}^k \in \{0,\ 1\} \qquad i = 0,...,n;\ j = 0,...,n;\ k = 1,...,m$$

Finally equation (9) guarantees the decision variables x_{ij}^k to be binary.

4. Scatter search overview

Scatter search (SS) is an instance of evolutionary methods, because it violates the premise that evolutionary approaches must be based on randomization – though they likewise are compatible with randomized implementations. Glover (1977, 1998) proposed the first description of the method and a scatter search template. SS operates on a set of reference solutions to generate new solutions by weighted linear combinations of structured subsets of solutions. It uses strategies for search diversification and intensification that have proved effective in a variety of optimization problems.
The following parameters are used in the method discussion:

$PSize$ = size of the set of diverse solutions generated by the Diversification Generation Method
b = size of the reference set (*RefSet*)
b_1 = size of the high-quality subset of *RefSet*
b_2 = size of the diverse subset of *RefSet*
MaxIter = maximum number of iterations

A sketch of the scatter search method is presented in figure 1 based on Alegre et al. (2004) and Yamashita et al. (2006). The first step (diversification generation method) generate a set *P* of diverse trial solutions, with *PSize* elements, using one or more arbitrary trial solutions as an input. The improvement method (step 2) is applied to transform a trial solution into one or more enhanced trial solutions. If no improvement occurs for a given trial solution, the enhanced solution is considered to be the same as the one submitted for improvement. Step 3 chooses b solutions from P, according to their quality or diversity, in order to build the reference set (*RefSet*), which is a collection of both high quality solutions and diverse solutions. In the step 4, solutions are combined. For each combined solution we apply the improvement method (step 5). At this point, the reference set can be updated (step 6), depending on the quality or diversity of the new combined solution. In this work, dynamic and static *RefSet* updating are implemented, as described in section 5.3. If the search converges, i.e., no new solutions are found for inclusion in *RefSet*, then Step 7 rebuilds *RefSet*. The algorithm stops when a termination criterion – the maximum number of iterations, *MaxIter*, is reached.

Step 1: Generate solutions – generate a set *P* of *Psize* diverse trial solutions through the diversification generation method.
Step 2: Improve solutions – apply the improvement method to improve solutions generated in Step 1.
Step 3: Build the reference set: Put b_1 best solutions and b_2 diverse solutions *P* in the reference set (*RefSet*). Number of iterations = 0.
NewSolutions = TRUE
While (number of iterations < *MaxIter*) **do**
  While *NewSolutions* in *RefSet* **do**
    Step 4: Combine solutions
    Step 5: Improve solutions – Apply the improvement method for each combined solution.
    Step 6: Update reference set
  End while
  Step 7: Rebuild reference set: Remove the worst b_2 solutions from the *RefSet*. Generate *PSize* diverse trial solutions and apply the improvement method (step 1 and 2). Choose b_2 diverse solutions and add them to *RefSet*. Number of iterations = Number of iterations + 1.
End while

Figure 1. Scatter search algorithm

5. Solution method

The initial reference set is composed of solutions generated using the constructive heuristic described at section 5.1. The scatter search procedure to solve the VRPTWSD is presented at section 5.2.

5.1 Adaptation of sequential insertion's heuristic of Solomon (1987)

This is an extension of Solomon's sequential insertion heuristics I1 (1987), although, in order to generate split deliveries, the vehicle capacity constraint must not be respected, so that the demand of a customer is added while there is capacity. The initialization criterion of the route is the farthest customer yet not allocated. The insertion criterion aims to minimize the addition of distance and time caused by a customer's insertion.
The heuristics begins with the farthest customer yet not allocated (customer *i*). If the demand of customer *i* is larger than the vehicle capacity, a vehicle with full truckload is sent and the remaining demand is added to a new vehicle. Next step is the insertion of a new customer *j* to the route. If the total demand of the customer *j,* plus the demand of customer *i,* exceeds the capacity of the actual vehicle, the demand of customer *j* is added while there is still capacity, and the remaining demand is added to a new vehicle. This process goes on until all the customers belong to a route.

5.2. Scatter search procedure

This section describes the scatter search procedure proposed to solve the vehicle routing problem with time windows and split deliveries.

Diversification generation method

The initial population must be a wide set of disperse individuals. However, it must also include good individuals. So, individuals of the population are created by using a random procedure in the constructive heuristics to achieve a certain level of diversity. In the next step, an improvement method must be applied to these individuals in order to get better solutions.

So, a randomized version of the constructive heuristic, called H-random, may be achieved by modifying the initialization criterion and the insertion criterion. In H-random, customer j is randomly selected from a candidate list. The candidate list is created by first defining *r*_first and *r*_last as the first and last customer in the list. If the initialization criterion considers the customer with the earliest deadline, *r*_first and *r*_last are the customers with the earliest and latest deadline, respectively. If the initialization criterion considers the customer farthest from the depot, *r*_first and and *r*_last are the customers farthest and nearest from the depot, respectively. In the insertion criterion, *r*_first and *r*_last are the customers with the cheapest and more expensive insertion cost, respectively. A possible customer *i* is added to the list if

$$r_i \leq r\_first + \alpha(r\_first - r\_last)$$

where $\alpha(0 \leq \alpha \leq 1)$ is the parameter that controls the amount of randomization permitted. If α is set to a value of zero, H-random becomes the deterministic procedure described in subsections 5.1. On the contrary, the candidate list reaches its maximum possible cardinality when α is set to a value of one.

Improvement method

To each one of the *PSize* solutions obtained through the diversification generation method we apply the improvement method that is composed by 5 phases: swap in the same route, demand reallocation, route elimination and combination, insertion and route addition. The first exchange procedure is once again applied at the end of the process.

Swap in the same route

This exchange procedure has the goal of reducing the length of a route. The procedure is applied to each route R_k, for $k = 1,...,m$. From $i = 1,...,n_k - 1$ and $j = i+1,...,n_k$, the procedure tests the reduction of the length of a route R_k produced by exchanging the positions of $r_k(i)$ and $r_k(j)$. If the length is reduced and the constraints are respected, the positions are exchanged. The procedure stops when no more feasible exchanges are possible that result in a shorter route length.

Demand reallocation

This procedure is applied for each customer *i*, $i = 1,...,n$ which is split into more than one route. We begin with the farthest customer *i* from the depot. For a determined customer *i*, we initially choose the route R_j which deliveries the most quantity for customer *i*, because customers with superior demands have a larger probability of violating the capacity constraint if they are not inserted at the beginning (Salhi and Rand, 1993). Therefore, we calculate the demand reallocation cost of customer *i* of route R_j for each one of the other routes R_k where customer *i* is inserted and we choose the cheapest one. The exchange is done only if all the problem constraints are respected and the total cost reduced. The procedure is repeated for the other routes where customer *i* is inserted.

Route Elimination and Combination

This exchange procedure aims to eliminate routes with n or less customers and routes with idle capacity. For each possible route R_j eliminated (we begin with the longest route, that is, the one with the maximum length), we aim to combine it to another route R_k, chosen though a candidate list. This candidate list is based on Corberán et al. (2002) and is described below.

For each one of the routes evaluated (R_j), there is a candidate list, based on the best pairs of routes (R_j , R_k). The best pair is the one with the minimum distance between two routes. When the routes have only one customer, the distance between the routes is simply the distance between two customers. When a route has more then one customer, we consider the extreme points to calculate the minimum distance. The minimum distance between routes R_j and R_k is given by:

$$\min\left\{d_{r_j(1)r_k(n_k)}, d_{r_k(1)r_j(n_j)}\right\}$$

where:

$d_{r_j(1)r_k(n_k)}$ is the distance between the first customer on route R_j to the last one on route R_k

$d_{r_k(1)r_j(n_j)}$ is the distance between the first customer on route R_k to the last one on route R_j.

For each route R_j, we randomly choose a candidate route R_k from the candidate list. We attempt to merge the routes, considering R_j first and then R_k. If the merging is feasible, we stop and go to the next route R_j. A feasible merging of routes R_j and R_k is such that the resulting route does not violate the capacity and time windows constraints. If the merging is not feasible, we choose another candidate route R_k from the candidate list, while it will have a candidate route R_k.

Insertion

This exchange method is based on removing a customer from one route and inserting it into another route. The insertion movement is implemented for each route R_j, where all the feasible customers will be tested in all positions of another route R_k, chosen through a candidate list, similar to the one described in the routes elimination and combination procedure. We begin with the longest route R_j, which is the one with the maximum length, and for each route R_j chosen we begin with the farthest customer i from the depot. We select the best insertion position of customer i in route R_k. The customer is only inserted if the cost is reduced and all the constraints are respected.

Routes addition

Like in demand reallocating, we consider all customers i whose demand is split into more than one route. We begin with the farthest customer i from the depot. For each customer i, we choose the route R_j that deliveries the smallest quantity for customer i. In the next step, we relocate the demand of customer i from route R_j to a new route R_k and calculate the reduction or addition to the total cost after this movement. This process is repeated for the other routes where customer i is inserted. The demand is added to the actual route R_k while there is space and the remaining demand is added to a new vehicle. In the end, the best combination is chosen, if there is any saving.

Reference set update method

Solutions are included in the reference set by quality or diversity. The subset of quality solutions (*RefSet*$_1$) contains the b_1 best solutions and the subset of diverse solutions (*RefSet*$_2$) contains the b_2 diverse solutions. The initial *RefSet* consists of b_1 best solutions that belong to *P* and b_2 elements from P that maximize the minimum distance to *RefSet*. The distance between two solutions is calculated by adding the number of non-common arcs of each solution before the combination. If an arc belongs to more than one route, we add one unit for each non-common arc. We consider arc x_{i0} or x_{0i} only for routes with full truckloads $(0-i-0)$.

There are two main aspects that must be considered when updating the *RefSet*. The first one refers to the timing of the update. There are two possibilities: static update (S) and dynamic update (D). The second aspect deals with choosing the criteria for adding to and deleting elements from *RefSet*. We consider two types of updates: by quality (Q) and by quality and diversity (QD).

In the static update (S), the reference set doesn't change until all solutions combinations of *RefSet* were done. In the dynamic update (D), the reference set is updated when a new better solution is found.

In the quality update (Q), if a new solution is better than the worst element of *RefSet*$_1$, then the worst element of *RefSet*$_1$ is substituted by the new solution. In the quality and diversity update (QD), if a new solution is better than the worst element of *RefSet*$_1$, then the worst element of *RefSet*$_1$ is substituted by the new solution. If a new solution is worse than the worst element of *RefSet*$_1$, but it increases the minimum distance between the solutions in *RefSet*$_2$, the solution with the minimum distance in *RefSet*$_2$ is substituted by the new solution.

Solution combination method

The solution combination method is applied to all pairs of solutions in the current referent set.

This method is divided into two steps. Step 1 aims to combine only the common elements of the combined routes and step 2 supplies the remaining demand of the customers. The combination method was based on the ideas of Corberán et al. (2002) and Rego and Leão (2000).

Step 1

Let A be a solution with m routes and B a solution with n routes, where A_i is the *i*-th route for solution A, $i = \{1,...,m\}$, and B_k is the *k*-th route for solution B, $k = \{1,...,n\}$. The solutions A and B are combined. The combination procedure of step 1 is built from a matrix $A \times B$, where its component (A_i, B_k) have the number of common elements between the route *i* of solution *A* and route *k* of solution *B*. Firstly we define which routes are combined, that is, which component (A_i, B_k) is chosen. It begins with the component with a greater number of common elements. If there is a tie, the combination that minimizes the function *f*, $f = \sum_{j \in i,k} \left| y^A_{j,i} - y^B_{j,k} \right|$ is chosen, where *j* is the common element of the route i of solution *A* and of the route k in the solution *B*; $y^A_{j,i}$ is the quantity delivered to customer *j* from route i in solution *A* and $y^B_{j,i}$ is the quantity delivered to the customer *j* from route k in solution *B*.

The combined route is formed by the routes' common elements. The quantity delivered to each element is the smallest quantity between two solutions combined. The route of the combined solution is similar to the one of the best solution and, if there is a tie, we choose randomly the route of one of the solutions. Each combined route is excluded from the list (we delete the line and column referring to components A_i, B_k). The procedure follows while there are routes (which have not been combined yet) with common elements.

Step 2

This step aims to fulfill the remaining demand of customers who have already been served by some route in step 1, and fulfill the complete demand of customers who still do not belong to any route. This is done through an insertion procedure. Customer i is the farthest one from the depot, and is going to be inserted. First it is verified if it already belongs to any route combined, and after that two steps can be taken:

Step 2.1

If the customer i already belongs to at least one combined route (step 1 of the combination method), the one with larger idle capacity is chosen and it is delivered the minimum between the idle capacity of the vehicle and the customer demand. While the total customer demand i is not fulfilled and there is a route with idle capacity, in which the customer i is inserted, this procedure is repeated.

By the end of this step, if the total demand of customer i has not been fulfilled, we go to step 2.2. Otherwise, the next farther customer is chosen and inserted.

Step 2.2

If customer *i* does not belong to any combined route or the total demand of the customer i has not been fulfilled through step 2.1, step 2.2 is taken. Based on the initial solutions *A* and *B* (before combination), all the routes in which customer *i* is inserted are verified and also all the arcs in which $x_{ij} = 1$ (customer *i* is attended before customer *j*) or $x_{ji} = 1$ (customer *i* is attended after customer *j*). Therefore, for each *j* belonging to one of the combined routes in the step 2.1, except the depot $(j = 0)$, we calculate the cost for inserting customer *i* (addition of fixed cost, routing cost and time) before customer *j* (when $x_{ij} = 1$) or the cost for inserting customer *i* after customer *j* (when $x_{ji} = 1$). It is considered $x_{i0} = 1$ or $x_{0i} = 1$ only for routes with full truckload $(0 - i - 0)$.

For each possible position of customer insertion *i*, we deliver the minimum between the idle capacity of the vehicle and the demand of customer *i*, and we choose the one with the minimum cost, since the time window constraint has been respected. The procedure is repeated until the total demand of the customer *i* is fulfilled or while there is a position to insert customer *i*.

Once all the routes from solutions *A* and *B* where customer *i* is inserted have been verified, if the total demand has not been fulfilled, we add a new route. The procedure is repeated until the total demand of customer *i* is fulfilled.

The combination method guarantees the feasibility of the final solutions. For each combined solution we apply the improvement method.

The algorithm stops when $MaxIter = 5$ or when reaching the maximum time of 1 hour, until the last iteration is finished.

6. Experimental results

6.1 Problem sets

The Solomon's problem set consists of 100 customers with Euclidean distance. The percentage of customers with time windows varies between 25, 50, 75 and 100% according to the problem. The author considers six sets of problems: R1, R2, C1, C2, RC1 and RC2. In sets R1 and R2 the customers' position is created randomly through a uniform distribution. In sets C1 and C2 the customers are divided in groups. In sets RC1 and RC2, customers are in sub-groups, that is, part of the customers is placed randomly and part is placed in groups. Besides, R1, C1 and RC1 problems have got a short term planning horizon and, combined to lighter capacity vehicles, they allow only some customers (3-8) in each route. Sets R2, C2 and RC2 have got a long term planning horizon and, since they have got higher capacity vehicles, they are able to supply more than 10 customers per route.

In each set of problems, the customers' geographical distributions, the demand and the service time do not change. Therefore, on set R1, the problems from R101 to R104 are identical, except for the customer with time window percentage, which is 100% in problem R101, 75% in problem R102, 50% in problem R103 and 25% in problem R104. Problems from R105 to R108 are identical to problems from R101 to R104, the only difference is the time window interval. The same occurs for problems R106 to R112.

Ho and Haugland (2004) have changed the demands from Solomon's problems, aiming to allow more than one delivery per customer. We consider m the vehicle capacity and w_i the customer i's demand $i=1,\ldots, n$. Each demand is recalculated within the period $[lm, um]$, where $l < u$ are defined within the period [0, 1]. Therefore, for every $i \in C$, we define a new demand $w_i' = lm + m((u-l)/(\overline{w}-\underline{w}))(w_i - \underline{w})$, where $\underline{w} = \min\{w_i : i \in C\}$ and $\overline{w} = \max\{w_i : i \in C\}$. The new demand w_i' is evened to the closest integer number.

For each problem set the authors use the following values of $[l, u]$:

l	u
0,01	0,50
0,02	1,00
0,50	1,00
0,70	1,00

Using this method, sets totaling 224 problems were created.

Table 1. Demand Values

6.2 Experiments on Solomon's problems with augmented demands

Initially, many different values for b_1, b_2, $PSize$, updating criteria and frequency were tested. The best results obtained were: $PSize = 30$, $b_1 = 5$, $b_2 = 5$, static (S) and quality and diversity (QD) update.

Table 2 compares the results of the algorithm presented in this chapter with the results from Ho and Haugland (2004), for each class of problems, considering different [l, u] values.

l, u		R1	C1	RC1
0.01, 0.50	VRPTWSD$^{H\&H}$	18.25/1471.49	12.22/1182.12	20.13/1965.05
	VRPTWSD$^{B\&Y}$	18.42/1475.54	**12.22/1160.74**	21.00/**1941.25**
0.02, 1.00	VRPTWSD$^{H\&H}$	35.00/2291.46	22.22/2168.57	40.00/3339.20
	VRPTWSD$^{B\&Y}$	35.83/2302.58	24.00/**2009.37**	41.75/3425.96
0.50, 1.00	VRPTWSD$^{H\&H}$	67.00/4040.67	61.00/3979.78	70.00/5453.10
	VRPTWSD$^{B\&Y}$	69.50/**4035.84**	**60.75/3975.49**	73.75/**5231.85**
0.70, 1.00	VRPTWSD$^{H\&H}$	79.00/4581.54	77.00/4962.28	81.00/6095.20
	VRPTWSD$^{B\&Y}$	82.75/**4464.85**	**76.88/4950.81**	82.50/**6013.92**
l, u		R2	C2	RC2
0.01, 0.50	**VRPTWSD$^{H\&H}$**	**18.00/1430.62**	11.13/1174.29	20,00/1946,07
	VRPTWSD$^{B\&Y}$	18.00/1425.40	11.75/1180.34	21.00/1941.42
0.02, 1.00	**VRPTWSD$^{H\&H}$**	35.00/2318.04	22.00/1995.59	39,00/3419,85
	VRPTWSD$^{B\&Y}$	35.82/**2314.65**	23.13/**1993.47**	41.50/**3410.65**
0.50, 1.00	VRPTWSD$^{H\&H}$	68.00/4059.26	61.00/4268.02	71,00/5546,20
	VRPTWSD$^{B\&Y}$	69.27**/4055.29**	**60.88/4259.14**	**71.00/5498.32**
0.70, 1.00	VRPTWSD$^{H\&H}$	80.00/4574.17	77.00/5246.12	82,00/6155.49
	VRPTWSD$^{B\&Y}$	82.82/4625.87	**76.88/5214.79**	83.25/6217.43

Legend
VRPTWSD$^{H\&H}$: Ho and Haugland's VRPTWSD results (2004)
VRPTWSD$^{B\&Y}$: Belfiore and Yoshizaki's VRPTWSD results (2008)

Table 2. Comparison to Ho and Haugland (2004) results for Solomon's modified demands.

According to table 2, we can conclude that, in 6 classes of problems, the scatter search metaheuristic's average has overcome the best results from Ho and Haugland (2004). Besides, in other 11 problems, it was possible to reduce the average distance traveled, but the average number of used vehicles was higher. It has also been verified that the best results were found for $[l,u]=[0.50,\ 1.00]$ e $[0.70,\ 1.00]$, because higher demand values increases the possibility of split deliveries.

Table 3 shows the average processing time (seconds), for the classes of problems R1, C1, RC1, R2, C2 and RC2, considering different values of [*l, u*].

l, u	**R1**	**R2**	**C1**	**C2**	**RC1**	**RC2**
0.01, 0.50	807	982	517	1121	678	1030
0.02, 1.00	840	1022	785	888	905	926
0.50, 1.00	1025	1014	574	876	1014	885
0.70, 1.00	812	899	755	922	901	957

Table 3. Average processing time (seconds) for Ho and Haugland (2004) modified demands.

7. Conclusions and future research

We have proposed a scatter search algorithm for the Vehicle Routing Problem with Time Windows and Split Deliveries. The initial solution is an extension of Solomon's sequential insertion heuristic I1. The scatter search framework provided a means to combine solutions, diversify, and intensify the metaheuristic search process.

The algorithm was applied on Solomon's problems with augmented demands. For the problems sets, many parameters of scatter search metaheuristic was tested: *PSize*, b_1, b_2, updating criteria and updating frequency.

Computational testing revealed that our algorithm matched some of the best solutions on the Solomon's problem set with augmented demands. It has also been verified that the best results were found for $[l,u]=[0.50,\ 1.00]$ e $[0.70,\ 1.00]$, because with higher demand values higher is the possibility of split delivery.

As future research, other constructive heuristic can be applied as an initial solution. Other improvement heuristics can also be tested. Besides, the Scatter Search metaheuristic proposed can be adapted for other problems, as VRP with heterogeneous fleet, multiple depots, etc.

8. Acknowledgments

This research was partially funded by CAPES. The authors also thank Vanzolini Foundation for giving support to presentations.

9. References

Archetti, C, M W P Savelsbergh and M G Speranza (2003a), Worst-Case Analysis of Split Delivery Routing Problems, Department of Quantitative Methods, University of Brescia and School of Industrial and Systems Engineering, Georgia Institute of Technology, Accept in *Transportation Science*.

Archetti, C, A Hertz and M G Speranza (2003b), A Tabu Search Algorithm for the Split Delivery Vehicle Routing Problem, Les Cahiers du GERAD, Accept in *Transportation Science*.

Archetti, C, M Mansini and M G Speranza (2005), Complexity and Reducibility of the Skip Delivery Problem, *Transportation Science* 39, 182-187.

Belenguer, J M, M C Martinez and E Mota (2000), A Lower Bound for the Split Delivery Vehicle Routing Problem. *Operations Research* 48, 801-810.

Belfiore, P P (2006), Scatter Search para Problemas de Roteirização de Veículos com Janelas de Tempo e Entregas Fracionadas. Tese (Doutorado), Universidade de São Paulo.

Corberán, A, E Fernández, M Laguna, R Martí (2002), Heuristic solutions to the problem of routing school buses with multiple objectives, *Journal of the Operational Research Society* 53, 427-435.

Dror, M and P Trudeau (1989), Savings by Split Delivery Routing, *Transportation Science* 23, 141-145.

Dror, M and P Trudeau (1990), Split Delivery Routing, *Naval Research Logistics* 37, 383-402.

Dror, M, G Laporte and P Trudeau (1994), Vehicle routing with split deliveries, *Discrete Applied Mathematics* 50, 229-254.

Feo, T A and M G C Resende (1989), A probabilistic heuristic for a computationally difficult set covering problem, *Operations Research Letters* 8, 67-71.

Frizzell, P W and J W Giffin (1992), The bounded split delivery vehicle routing problem with grid network distances, *Asia Pacific Journal of Operational Research* 9, 101-106.

Frizzell, P W and J W Giffin (1995), The Split Delivery Vehicle Scheduling Problem with Time Windows and Grid Network Distances, *Computers & Operations Research* 22, 655-667.

Glover, F (1977), Heuristics for Integer Programming Using Surrogate Constraints, *Decision Sciences* 8, 156-166.

Glover, F (1998), A Template for Scatter Search and Path Relinking, in *Artificial Evolution*, Lecture Notes in Computer Science 1363, Hao, J.-K., E. Lutton, E. Ronald, M. Schoenauer and D. Snyers (Eds.), Springer-Verlag, p13-54.

Ho, S C and D Haugland (2004), A tabu search heuristic for the vehicle routing problem with time windows and split deliveries, *Computers & Operations Research* 31, 1947-1964.

Lenstra, J K and A H G Rinnooy Kan (1981), Complexity of Vehicle and Scheduling Problems, *Networks* 11, 221-227.

Mullaseril, P A, M Dror and J Leung (1997), Split-delivery Routing Heuristics in Livestock Feed Distribution, *Journal of the Operational Research Society* 48, 107-116.

Rego, C and P A Leão (2000), Scatter Search Tutorial for Graph-Based Permutation Problems, Hearin Center for Enterprise, University of Mississipi, HCES-10-00, USA, in *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*, Rego, C and B Alidaee, B (Eds.), Kluwer Academic Publishers, p1-24, 2005.

Salhi, S and G K Rand (1993), Incorporating vehicle routing into the vehicle fleet composition problem, *European Journal of Operational Research* 66, 313-330.

Solomon, M M (1987), Algorithms for the Vehicle Routing and Scheduling Problems with Time Windows Constraints, *Operations Research* 35, 254-265.

Solomon, M M and J Desrosiers (1988), Time Window Constrained Routing and Scheduling Problem, *Transportation Science* 22, 1-13.

2

A Modelling and Optimization Framework for Real-World Vehicle Routing Problems

Tonči Carić[1], Ante Galić[1], Juraj Fosin[1], Hrvoje Gold[1] and Andreas Reinholz[2]
[1]Faculty of Transport and Traffic Sciences, University of Zagreb
[2]TU Dortmund
[1]Croatia,
[2]Germany

1. Introduction

The globalisation of the economy leads to a rapidly growing exchange of goods on our planet. Limited commodities and transportation resources, high planning complexity and the increasing cost pressure through the strong competition between logistics service providers make it essential to use computer-aided systems for the planning of the transports. An important subtask in this context is the operational planning of trucks or other specialized transportation vehicles. These optimization tasks are called Vehicle Routing Problems (VRP). Over 1000 papers about a huge variety of Vehicle Routing Problems indicate the practical and theoretical importance of this NP-hard optimization problem. Therefore, many specific solvers for different Vehicle Routing Problems can be found in the literature. The drawback is that most of these solvers are high specialized and inflexible and it needs a lot of effort to adapt them to modified problems. Additionally, most real world problems are often much more complex than the idealized problems out of literature and they also change over time. To face this issue, we present an integrated modelling and optimization framework for solving complex and practical relevant Vehicle Routing Problems. The modular structure of the framework, a script based modelling language, a library of VRP related algorithms and a graphical user interface give the user both reusable components and high flexibility for rapid prototyping of complex Vehicle Routing Problems.

1.1 Vehicle routing problem

The problem of finding optimal routes for groups of vehicles, the Vehicle Routing Problem (VRP), belongs to the class of NP-hard combinatorial problems. The fundamental objectives are to find the minimal number of vehicles, the minimal travel time or the minimal costs of the travelled routes. In practice the basic formulation of the VRP problem is augmented by constraints such as e.g. vehicle capacity or time interval in which each customer has to be served, revealing the Capacitated Vehicle Routing Problem (CVRP) and the Vehicle Routing Problem with Time Windows (VRPTW) respectively. The real-world problems mostly encompass the capacity and time constraints. For solving VRPTW problems, a large variety of algorithms has been proposed. Older methods developed for the VRPTW are described in

the survey (Cordeau et al., 2002) and (Laporte, 1992). Most of the new methods tested on Solomon's benchmarks are comprised in (Bräysy & Gendreau, 2005a; Bräysy & Gendreau, 2005b). The methods that applied the two-phase approach for solving VRPTW are found to be the most successful (Bräysy & Dullaert, 2003). During the first phase the constructive heuristic algorithm is used to generate a feasible initial solution. In the second phase an iterative improvement heuristics is applied to the initial solution. The mechanism for escaping the local optima is often implemented in the second phase, too.
For the real-world application that solves VRP it is essential to perform a fast selection of methods (constructive heuristics, neighbourhood operators and escaping mechanisms) which produce the desired improvement of the objective function. Commercial VRP applications mostly converge to self-adaptive procedures with major aim of robustness to solve the problem with minimal human intervention in algorithms tuning. On the other side tailor-made solution needs easy-to-use prototyping tool with the well defined performance measure for estimating the optimal number of restarts and iterations of the implemented algorithms. To speed up the prototyping a new VRP framework has been developed.

2. Framework

The framework consists of the Framework Scripting Language (FSL), library of VRP-related algorithms and graphical user interface which enables loading of standard benchmarks and real-world problems, Fig. 1.

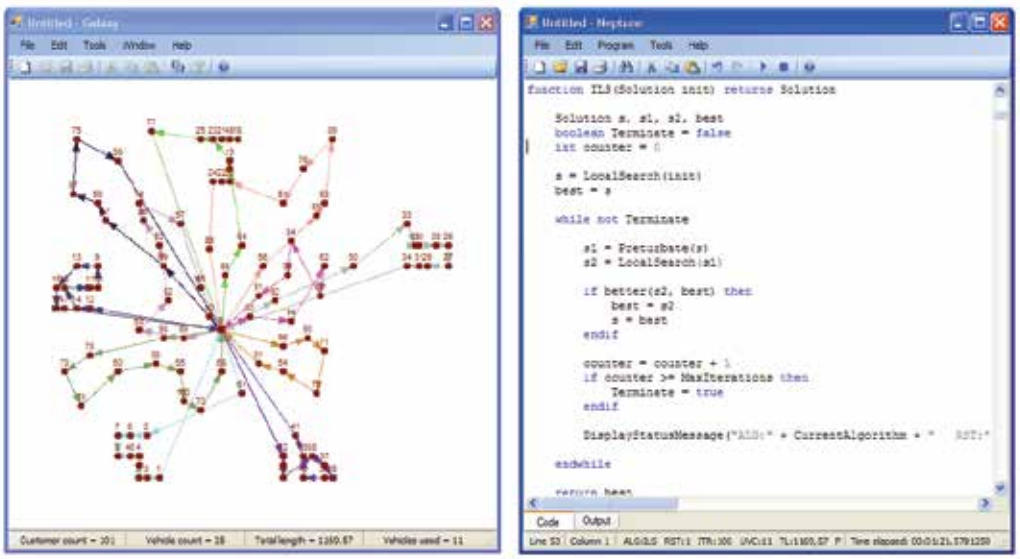


Fig. 1. VRP working environment and Framework Scripting Language

The VRP framework provides the working environment and the reusable code modules such as constructive heuristic methods and common improvement operators. This programming environment leaves more time for the developer to focus on the implementation and testing of the new ideas. The whole library is written in the framework language so that the included programming modules can be easily adapted to the needed functionality.

2.1 Framework scripting language

Like many other programming languages, the Framework Scripting Language (FSL) has a core set of basic data types (boolean, int, double, string) and program control statements (if, for, while, repeat). The FSL structure is the improved version of the previously developed VRP solving oriented language (Galić et al., 2006a). The VRP problem is described with *Problem* data structure which stores all customers in the list *Customers* and all vehicles in the list *Vehicles*. Each *Customer* and *Vehicle* is instances of the corresponding data type whose attributes describe in detail the concrete problem being solved. The VRP-related data types and the corresponding attributes and methods which are available for use in FSL are depicted in Fig. 2. Every solution of the VRP problem can be stored in an instance of *Solution* data type which holds the routes of vehicles (visiting order of customers) and other information which describe specific solution such as the number of used vehicles, total travelled distance and total estimated time which is the sum of the driving, serving and waiting time for each used vehicle. Using the *SetCurrentSolution* method it is possible to switch between different solutions of the same VRP problem.

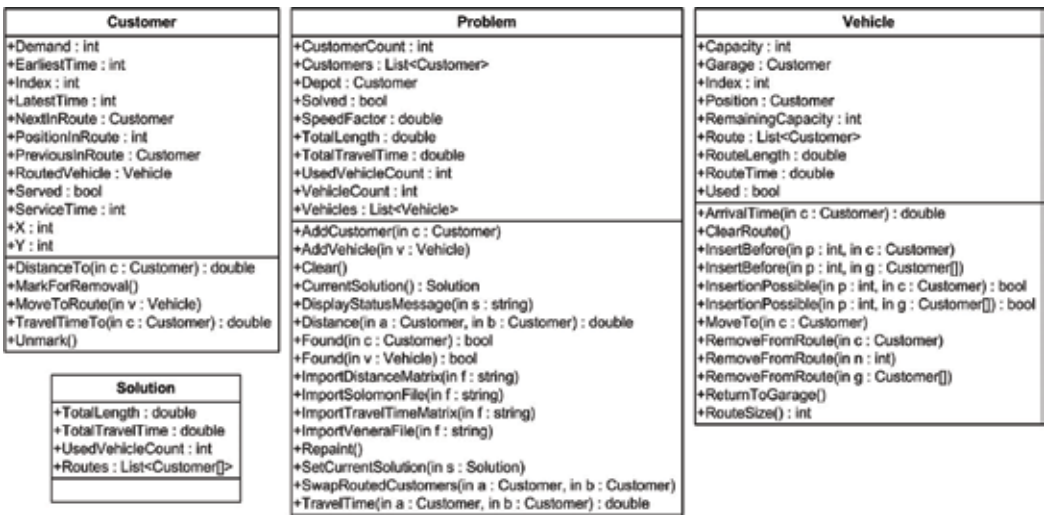


Fig. 2. VRP Framework Scripting Language data types, attributes and methods

2.2 Library

The library includes a variety of constructive heuristics, e.g. Clark and Wright, Solomon Insertion I1, Coefficient Weighted Distance Time Heuristic and neighbourhood operators, e.g. relocate, exchange, cross exchange. The included examples of procedures for escaping the local optima like Simulated Annealing, Iterated Local Search and Variable Neighbourhood Search present the proposed programming style of modules 'gluing'. VRP solver produced by this prototyping tool is an algorithm which can be composed by choosing and tuning modules from a library and that guides the search through local optima to achieve a better solution.

2.2.1 Constructive heuristics

The first step of the heuristic VRP solving is the construction of a feasible initial solution. In the lucky case when handmade solution already exists, it can be considered as a substitution

for the constructive heuristics. Generally, constructive heuristics follows the idea that customers are selected on some cost minimization criterion and routes are constructed matching capacity and time constrains. Methods with sequential approach construct one route at a time, while parallel methods build several routes simultaneously. Some constructive methods are two-phase methods and can be divided into two classes: cluster-first, route-second methods and route-first, cluster-second methods. In the first case, customers are first organized into feasible clusters, and a vehicle route is constructed for each of them. In the second case, a tour is first built on all customers and then segmented into feasible vehicle routes.

2.2.1.1 Nearest Neighbour Heuristic for CVRP

The Nearest Neighbour Heuristic (NNH) is a constructive method for generating initial feasible solution for CVRP with the simple idea of inserting the nearest neighbour of the last inserted customer in the route. The first inserted customer on the route can be selected randomly or with some arbitrary criteria like the farthest distance customer from the depot. From this seed route, every other customer is inserted by the criteria of the nearest neighbour from the last inserted customer until the capacity of the vehicle is exhausted according to the definition of the CVRP problem where every customer has its own demand for the delivery or pick up. This method is derived from Travelling Salesman Problem (TSP) heuristic approach (Flood, 1956).

2.2.1.2 Nearest Addition Heuristic for CVRP

The Nearest Addition Heuristic (NAH) is an extended version of NNH where one of the unserved customers is selected for insertion and added to the existing route between two already visited neighbours. The total price of insertion has to be the minimal value that is calculated by adding two new distances produced by linking of the unvisited customer with neighbours and by subtracting the distance between the visited neighbour's customers in the selected route.

2.2.1.3 Sweep Heuristic for CVRP

One of the most known two-phase constructive methods for CVRP is the sweep algorithm (Gillet & Miller, 1974). This is a two-phase algorithm that belongs to the cluster-first, route-second methods. In the first phase the algorithm decomposes the CVRP problem by clustering customers in m-TSP problems. The customer clustering is conducted by two criteria. The positions of all customers are transformed in polar coordinates with the depot in the origin of the coordinate system. The first criterion for grouping customers is the minimal angle. The second criterion matches the capacity of the vehicle which is assigned to the cluster, so that the total demands of all the selected customers has to be less than or equal to the capacity of the vehicle. The first and the second criteria are combined so that the assignment of customers to groups is performed by increasing the angular coordinate from 0 to the value where capacity of the assigned vehicle for that cluster is exhausted. The last step optimizes each vehicle route (cluster) separately by solving the corresponding TSP.

2.2.1.4 Clark and Wright Heuristic for CVRP

This method is one of the first originally developed heuristics for CVRP and it is frequently used. The algorithm starts from the initial solution where each route has only one customer and a corresponding vehicle. At the start, the number of vehicles is equal to the number of customers. Every new iteration should reduce the number of vehicles unifying two routes that give maximal savings, e.g. reduction of overall distance or time. There are two variants of algorithm: one with sequential and other with parallel construction of routes. The parallel version yields better results (Clarke & Wright, 1964).

2.2.1.5 Solomon's Sequential Insertion Heuristic I1 for CVRPTW

The seed customer for a new route can be set on various criteria (e.g. the farthest unrouted customer, unrouted customer with the earliest deadline or unrouted customer with the biggest demand). For each unrouted customer the feasible insertion place in the emerging route with its minimal insertion cost as a weighted average of additional distance and time is computed first. The next step is to select a customer for whom the cost difference between insertion in a new route and in the emerging route is the largest. The selected customer is then inserted in the route and the new calculation and selection is repeated until the time or capacity resource is exhausted. New resources are generated with new route/vehicles. It is not trivial to find the suitable weighted average for real-world problems. A good starting point for the tuning algorithm can be found in the original paper (Solomon, 1987).

2.2.1.6 Coefficient Weighted Distance Time Heuristics for CVRPTW

Based on the assignment of weights to the closing part of a time windows and distances to the serving places, the Coefficient Weighted Distance Time Heuristics (CWDTH) has been developed (Galić et al., 2006b; Carić et al., 2007). In each iteration the algorithm simultaneously searches for the customer with the soonest closing time of requested delivery and minimum distance from the current vehicle position. The route is designed starting with one vehicle. In each subsequent iteration the customer who best matches the given criteria is served. When the vehicle has used all of the available capacity that can be utilized regarding the amount of demands, it returns to the depot. A new vehicle is engaged and the described process is repeated. At the moment when all customers have been served, the algorithm stops. Automatic parameter adjusting is implemented for the weighting of distance over delivery closing time.

2.2.2 Local search

The local search starts from the initial solution (e.g. provided by constructive heuristic method) and subsequently moves from the present solution to a neighbouring solution in the search space where each solution has only a relatively small number of feasible neighbour solutions and each of the moves is determined by neighbourhood's operators.

The library includes two groups of operators. Operators from the first group move one or more customers from one position in the route to another position in the same route and are called Intra Route operators, Fig. 3.

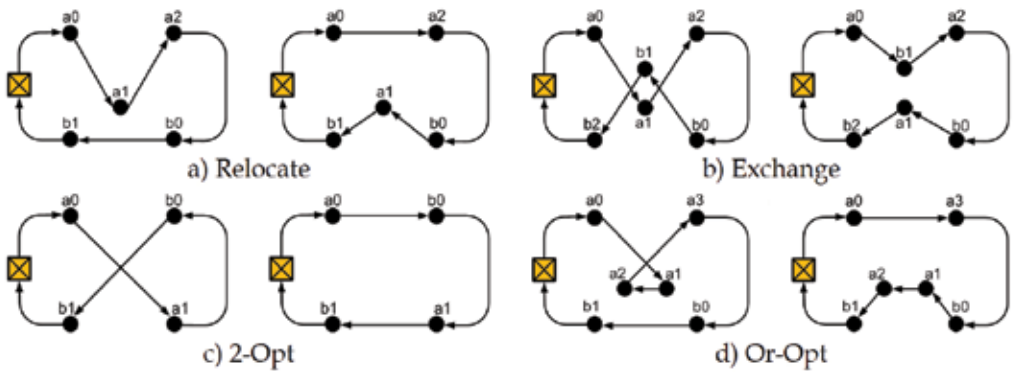


Fig. 3. Intra Route operators for CVRPTW

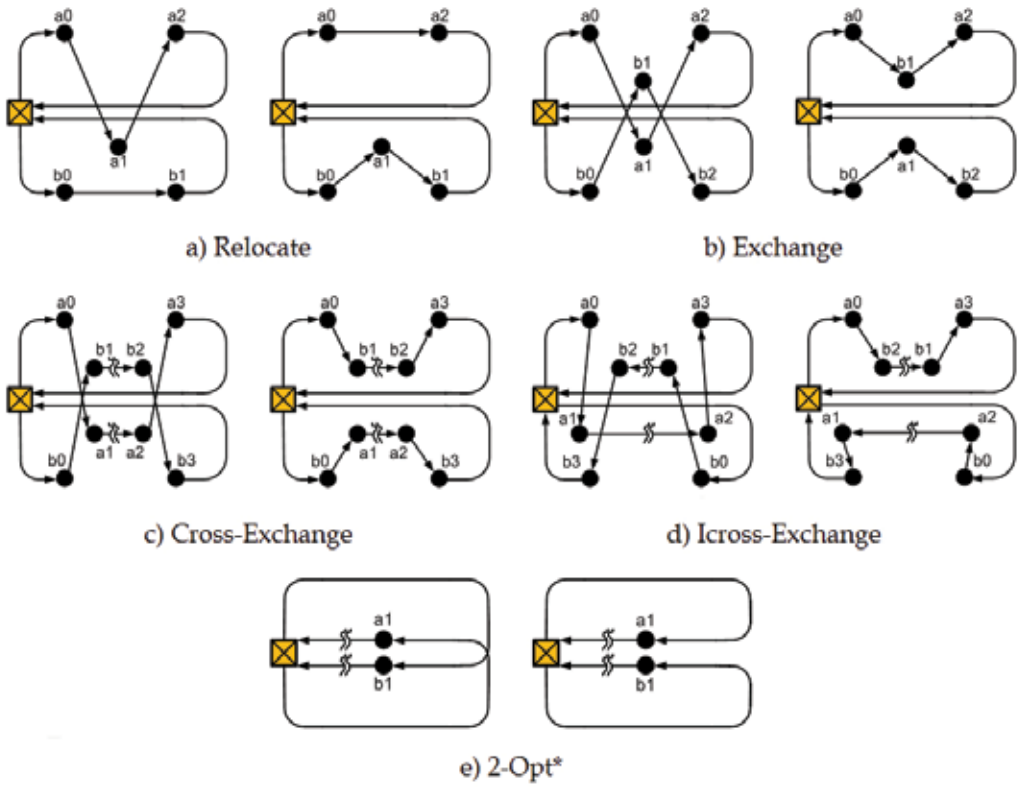


Fig. 4. Inter Route operators for CVRPTW

They are used for the reduction of the overall distance. The other group, called Inter Route operators, work with two routes, Fig. 4. They are used to reduce overall distance but in some cases they can reduce the number of vehicles as well.

2.2.2.1 Intra Route Relocate for CVRPTW

To be served in the new order between b0 and b1 the customer a1 is relocated from the original position between a0 and a2, (see Fig. 3a). Relocation is performed only when the saving (reduction of length of the route) is positive. The saving is calculated by maximizing the result of subtraction x-y where x is derived as the result of three arc deleting operations (a0, a1), (a1, a2) and (b0, b1) and y is derived as the result of three arc adding operations of (a0, a2), (b0, a1) and (a1, b1).

2.2.2.2 Intra Route Exchange for CVRPTW

Intra exchange operator swaps the position of two customers a1 and b1 (see Fig. 3b). To be served in the new order between a0 and a2 customer b1 is relocated from the original position between b0 and b2. Also, customer a1 is relocated from the original position between a0 and a2 to the new position between b0 and b2. The exchange is performed only when the saving (reduction of length of the route) is positive. The saving is calculated by maximizing the result of subtraction x-y as in the Intra Relocate operator. This operator could be considered as execution of two relocate operators, but sometime because of hard time windows the intermediate solution is not feasible.

2.2.2.3 Intra Route 2-Opt for CVRPTW

Intra route 2-Opt operator transforms the intersection of arcs if savings exist after we change the direction of the arcs between a1 and b0 and delete and add the appropriate arcs (see Fig. 3c).

2.2.2.4 Intra Or-Opt for CVRPTW

If savings exist, the intra route operator Or-Opt transforms the intersection of arcs with reordering customers on a route (see Fig. 3d). This operator is practical because it is very fast. The alternative slower scenario is two relocate operator execution.

2.2.2.5 Inter Route Relocate for CVRPTW

The inter route Relocate operator moves customer a1 from one route to another between b0 and b1 if savings exist (see Fig. 4a).

2.2.2.6 Inter Route Exchange for CVRPTW

The Exchange operator swaps two customers a1 and b1 from two different routes if savings exist (see Fig. 4b).

2.2.2.7 Inter Route Cross-Exchange for CVRPTW

The Cross-Exchange operator swaps two groups of customers from one route to another (see Fig. 4c). The groups consist of one up to maximally five customers. Bigger groups are inefficient mainly because of slow execution time. To prevent neighbourhoods from being interlaced (exchange and cross-exchange) only one group a1-a2 or b1-b2 can have only one customer in the group. In that case a1=a2 or b1=b2.

2.2.2.8 Inter Route Icross-Exchange for CVRPTW

The Icross-Exchange operator swaps two groups of customers the same way as Cross-Exchange but reverse in order of customers in both groups (see Fig. 4d). Further extension of Icross and Cross operator can be to leave the order of one group and to reverse the order of another.

2.2.2.9 Inter Route 2-Opt for CVRPTW*

Inter Route 2-Opt* can be considered like Cross-Exchange where b2 and a2 customers are depot (see Fig. 4e).

2.2.3 Escaping mechanism

By applying only the neighbourhood's operators in local search in most of the cases leads the optimization to the local optima where operators cannot yield better solutions any more. To escape from the local optima the escaping mechanisms such as Simulated Annealing, Iterated Local Search and Variable Neighbourhood Search are implemented and their scripts in FSL can be found in the Appendix.

2.2.3.1 Simulated Annealing

Simulated Annealing (SA) is a stochastic relaxation technique that finds its origin in statistical mechanics (Kirkpatrick et al., 1983; Cerny, 1985; Metropolis et al., 1953). Simulated Annealing uses stochastic approach to guide the search. The method allows the search to continue in the direction of the neighbour even if the cost function gives inferior results in that direction. The starting solution is obtained by constructive heuristics, described in section 2.2.1

2.2.3.2 Iterated local search

The local search process is started by selecting an initial candidate solution and then proceeds by iteratively moving from one candidate solution to the neighbouring candidate solution, where the decision on each search step is based on a limited amount of local information only. In Stochastic Local Search (SLS) algorithms, these decisions as well as the search initialization can be randomized (Hoos & Stützle, 2005). Generally, in the Iterated Local Search (ILS) two types of SLS steps are used (Laurenço & Serra, 2002). One step for reaching the local optima as efficiently as possible and the other step for efficiently escaping local optima.

2.2.3.3 Variable Neighbourhood Search

Another way to escape local optima is the idea that one solution which is a local optima for one neighbourhood generated by one operator need not be a local optima for another neighbourhood generated by some other operator. The procedure of Variable Neighbourhood Search (VNS) approach is to change the neighbourhood (operator) whenever local optima is reached. The starting neighbourhood is usually generated by the simplest operator in the pool of available operators. When currently selected operator does not produce improvement the next operator is selected and process continues. If any of the available operators produce an improvement, whole process starts again with first (simplest) operator from the pool. In case when neither one of the operators (including last one) produces improvement the shake move (perturbation) is executed to escape local optima. A new cycle starts again from first i.e. simplest operator. Function DoOperator (see Appendix) is called from VNS function to execute desirable operator by passing integer variable that represents index of operator.

2.3 Examples of using the framework scripting language

2.3.1 Solving the travelling salesman problem

In order to demonstrate the scope and the usage of FSL, a simple example of solving a TSP problem (one-vehicle VRP) by two algorithms from library is described, Fig. 5. The initial solution is calculated by the nearest neighbour constructive heuristic, described in section 2.2.1.1, and further improvements are done by simplified Intra Route Relocate operator, described in 2.2.2.1. In line 3, method *Clear()* deletes all the previous routes (if there are any) and prepares the problem for solving from scratch. In line 4, the initial solution construction is obtained by *TspNearestNeighbour* function call. From line 5 to line 7, the current solution is improved in a loop which breaks after the IntraRelocate operator has yielded no improvement. Inside of the *TspNearestNeighbour* method, the *select* statement in line 11 is used for finding one customer among all *Customers* that satisfies two conditions (customer is not *Depot* and not served at the moment) and minimizes the objective function. The result of the search is stored in the variable of *Customer* data type called *nearest*.

The objective function defined in line 14 represents the distance between the current position of vehicle *v* and candidate customer *nearest*. The objective function is calculated only for those *Customers* for which both conditions have been fulfilled and the selecting process is ended with the *Customer* which has the lowest value of objective function. In other words, this query (lines 11-15) selects the nearest customer to the current position of the vehicle which is not depot and which is not being served at the moment. Despite similar syntax the FSL statement *select* should not be confused with SQL SELECT statement. Statements *v.MoveTo(nearest)* and *v.ReturnToGarage()* are examples of methods that have

impact on the current solution of the active problem. By moving vehicle v to the nearest customer or its garage (default is depot), the state of the active problem solution will be changed. For example, by moving vehicle v to customer A, the vehicle route will be updated and the attribute of its position will gain value of A.

```
1       Vehicle truck = Vehicles[0]
2       boolean success
3       Clear()
4       TspNearestNeighbour(truck)
5       repeat
6          success = IntraRelocate(truck)
7       until not success
8       function TspNearestNeighbour(Vehicle v) returns nothing
9          Customer nearest
10         while true
11            select Customer nearest from Customers where
12               : nearest != v.Garage
13               : nearest.Served = false
14               minimize Distance(v.Position, nearest)
15            endselect
16            if not Found(nearest) then
17               break
18            endif
19            v.MoveTo(nearest)
20         endwhile
21         v.ReturnToGarage()
22      endfunction
23      function IntraRelocate(Vehicle v) returns boolean
24         double x, y, improvement
25         Customer a0, a1, a2, b0, b1
26         select Customer a1 from v.Route, Customer b1 from v.Route where
27               : a1.Index != Depot.Index
28               : b1.Index != Depot.Index
29               : abs (a1.PositionInRoute - b1.PositionInRoute) > 1
30               a0 = v.Route[a1.PositionInRoute-1]
31               a2 = v.Route[a1.PositionInRoute+1]
32               b0 = v.Route[b1.PositionInRoute-1]
33               x = (a0.DistanceTo(a1) + a1.DistanceTo(a2) +
                 b0.DistanceTo(b1))
34               y = (b0.DistanceTo(a1) + a1.DistanceTo(b1) +
                 a0.DistanceTo(a2))
35               improvement = x - y
36               : improvement > 0.000000001
37               maximize improvement
38         endselect
39         if not Found(a1) then
40               return false
41         endif
42         v.RemoveFromRoute(a1)
43         v.InsertBefore(b1.PositionInRoute, a1)
44         return true
45      endfunction
```

Fig. 5. Complete script of simple Travelling Salesman Problem solved in Framework Scripting Language

The statement *select* from line 26 to 38 execute a selective search for two customers a1 and b1 from the route of vehicle *v* with the aim of maximizing the objective function defined in line 37. Lines between *select* and *endselect* labelled by colon are search constraints. Unlabelled lines are regular statements which are executed for each iteration of search to update the variables used as part constraints or objective function. Constraints in lines 27 and 28 request that customers, a1 and b1, cannot be equal to depot. Constraint in line 29 does not allow the case when customers a1 and b1 are two neighbours on the route. The final constraint, defined in line 36 assures that every improvement has to be positive and it resolves the problem regarding acceptable error in comparison to the floating point numbers. The objective function is calculated only for those values of the required variables at which all the set conditions have been fulfilled. If one of the conditions is never to be satisfied, the search will be unsuccessful, and will result in non-initialized variables. In that case, after the statement *select* has been performed, the variables can be tested and the decision about the next action can be made. It should be noted that at the end of the search, the variables a1 and b1 acquire values for which the conditions are satisfied and for which the variable *improvement* yields maximal value. In other words, the results of the search are customer a1 and the position located before customer b1 on the same route for which we obtain maximal saving by performing the relocate operation. With the statements *v.RemoveFromRoute(a1)* and *v.InsertBefore(b1.PositionInRoute, a1)* the relocation is made.

2.3.2 Operators and guiding optimization

In order to demonstrate the proposed programming style of module "gluing" a simple local search mechanism coded in FSL is presented in Fig. 6. This local search is declared as *LocalSearch* function that returns the solution. The call of *LocalSearch* function preserves the

```
1     function LocalSearch(Solution s) returns Solution
2        Solution current = CurrentSolution()
3        SetCurrentSolution(s)
4        Solution improved = s
5        while(true)
6           SetCurrentSolution(Relocate())
7           SetCurrentSolution(Exchange())
8           SetCurrentSolution(Cross())
9           SetCurrentSolution(Icross())
10          SetCurrentSolution(TwoOptInter())
11          if better(CurrentSolution(), improved) then
12             intraRoute()
13             improved = CurrentSolution()
14          else
15             break
16          endif
17       endwhile
18       SetCurrentSolution(current)
19       return improved
20    endfunction
```

Fig. 6. Example of library components gluing for Local Search used in Simulated Annealing and Iterated Local Search escaping mechanisms

current state of the problem. Before the local search is started the problem solution is stored in the variable *current*, line 2, and restored at the end of the function, line 18. The initial

solution for *LocalSearch* function is passed by argument *s*, and the result of the search is returned, line 19, by variable *improved*. The local search is defined in *while* loop, line 5-17, which is stopped when the current solution obtained by the operators is not better than the one stored in variable *improved*, line 11. In the body of the loop, line 6-10, the operators Relocate, Exchange, Cross, Icross, TwoOptInter, one by one, try to improve the solution until local optima is reached. All of these operators return the first best feasible solution from their neighbourhoods. In line 12, the result is additionally improved by intra route operators Relocate, Exchange, 2-Opt and Or-Opt.

3. Performance measure protocol

The results of most heuristics and metaheuristics for Vehicle Routing Problems depends on the initialisation (i.e. starting solution, seed solution, etc.) of the algorithm, so that multiple starts with different initialisation can lead to better solutions than a single run. Additionally, the search process of most metaheuristics is influenced by explicit or implicit stochastic decisions (i.e. starting solution, selecting a candidate solution in a neighbourhood for the next iteration, mutation operators in Evolutionary Algorithms (EA), kick moves in ILS, shake moves in VNS, etc.). Therefore, we are using a performance measure for multi-start approaches that is able to handle both deterministic and stochastic algorithms (Reinholz, 2003). This performance measure is motivated by following question: How often do we have to run an algorithm with a concrete parameter setting so that the resulting solutions are equal or better than a requested quality threshold at a requested accuracy level (i.e. 90%, 95%, and 99%). The lowest number of runs that assures these requests is called multi-start factor (*MSF*).

DEF 1:

The multi-start factor *MSF* of an algorithm *A* with concrete parameters *P*, accuracy level *AL*, quality threshold *T*, and success probability *p* (*T*) is defined by

$$MSF(A, P, T, AL) := \min(k \in \mathbf{N} \text{ with } 1 - (1 - p(T))^k \geq AL)$$

The *MSF* multiplied by the average runtime of the fixed parameterized algorithm is the performance measure *PM* that has to be minimized.

DEF 2:

The performance measure *PM* of an algorithm *A* with concrete parameters *P*, average runtime *AvRT* (*A, P*), accuracy level *AL*, and quality threshold *T* is defined by

$$PM(A, P, T, AL) := MSF(A, P, T, AL) \times AvRT(A, P)$$

The estimation of the *MSF* in a statistical method is based on the fact that the success probability *p* of being better than the requested threshold quality in one run is Bernoulli-distributed. Therefore, we can use a parameterized maximum likelihood estimator to determine the success probability *p* for one run. This implies that the success probability for *k* runs (in *k* runs there is at least one successful run) is exactly $1 - (1 - p)^k$ and that the *MSF* for reaching a requested accuracy level *AL* can be easily computed using a geometrical distribution with success probability *p*. The accuracy of the estimation of *PM* and *MSF* depends on the number of runs that are used to estimate *AvRT* (*A, P*) and *p* (*T*).

Two important key parameters of iterative multi-start algorithms are the number of algorithm restarts and the maximal number of iterations. The statistic method for estimating the performance measure *PM* for a requested quality threshold and accuracy level can be

used in an elegant way to determine the best combination out of these two parameters by simply computing the *PM* for the intermediate results after each iteration and identifying the iteration with the best *PM* value.
This shows again the importance for the output of intermediate results of an algorithm when making an empirical investigation. Having done *R* runs of an algorithm for *I* iterations with the output of intermediate results, then you have also the data out of *R* runs for a statistical analysis of the algorithm with stopping criteria 1, 2, … , *I*.
In this paper we have used the statistical data out of 30 runs for each algorithm and the problem instance to estimate the success rates and the average runtimes for all the stopping criteria up to 100 iterations. The statistical analysis was applied to a series of combinations out of three accuracy levels and two quality thresholds. For the accuracy levels we have used the predefined values 90%, 95%, and 99%. The quality thresholds were chosen out of the data by the following procedure: The first quality threshold *T1* was defined by the quality value that was reached by the worst out of 25% of the best runs after 100 iterations. The second quality threshold *T2* was defined by the quality value that was reached by the worst out of 10% of the best runs.

4. Computational results

4.1 Benchmark results

The efficiency of VRP solver is usually measured by cumulative result of the Solomon benchmarks (Solomon, 1987). Three different scripts for ILS, SA and VNS are tested in order to check the relevance of the proposed Framework library and language, Table 1.

	R1	R2	R3	R4	R5	R6	CM	CPU
HG	12.08 1211.67	2.82 950.72	10.00 828.45	3.00 589.96	11.50 1395.93	3.25 1135.09	408 57422	P400 3 / 1.6
BC	12.08 1209.19	2.73 963.62	10.00 828.38	3.00 589.86	11.50 11389.22	3.25 1143.70	407 57412	P933 1 / 512
PR	11.92 1212.39	2.73 957.72	10.00 828.38	3.00 589.86	11.50 1387.12	3.25 1123.49	405 57332	P3000 10 / 2.4
MBD	12.00 1208.18	2.73 954.09	10.00 828.38	3.00 589.86	11.50 1387.12	3.25 1119.70	406 56812	P800 1 / 43.8
ILS	13.08 1192.87	3.27 936.25	10.00 828.38	3.00 589.86	12.88 1371.97	4.00 1073.73	442 56353	P2000D 5 / 9
SA	13.00 1193.51	3.27 933.82	10.00 828.38	3.00 589.86	12.88 1373.05	4.00 1067.13	441 56290	P2000D 5 / 9
VNS	12.92 1200.84	3.27 951.94	10.00 832.46	3.00 598.46	12.88 1379.63	4.00 1091.96	440 56934	P2000D 5 / 9

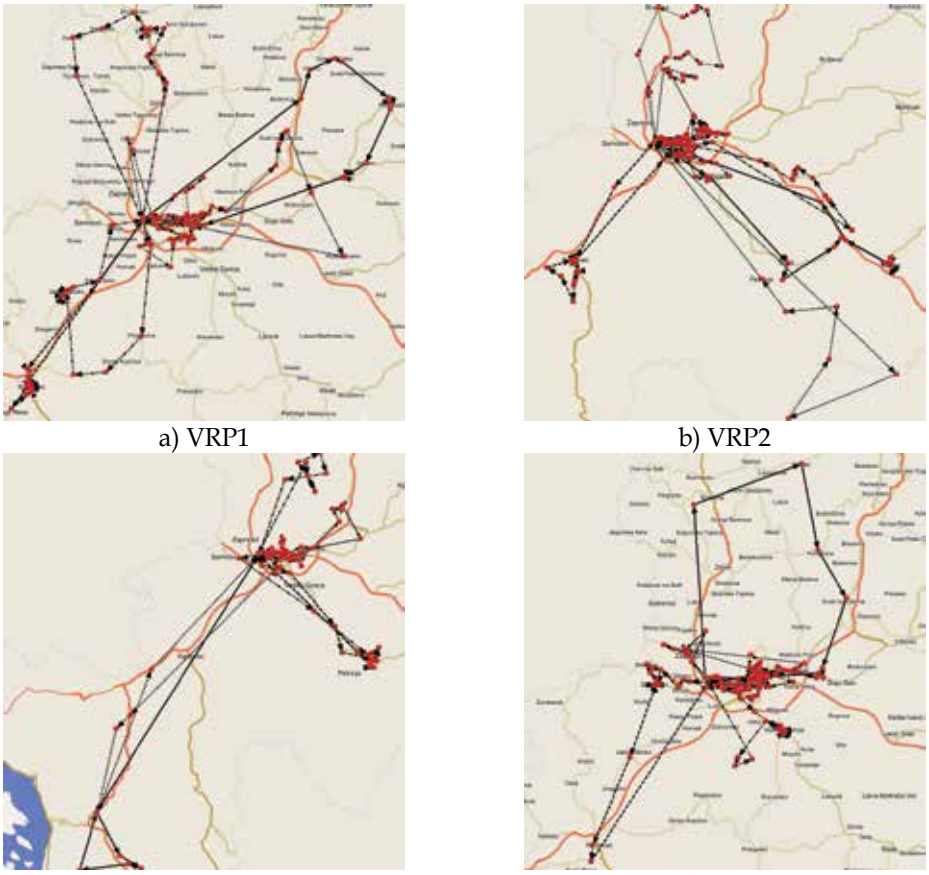
Table 1. Comparison of the results for the number of vehicles and distances obtained by ILS, SA and VNS to the best recently proposed results for Solomon's VRPTW problems. CM = cumulative values, P = Intel Pentium, D = duo, r / m = number of run(s) / minutes. HG = (Homberger & Gehring, 2005), BC = (Le Bouthillier & Crainic, 2005), PR = (Pisinger & Röpke, 2005), MBD = (Mester et al., 2007)

The modules (Solomon I1, Intra and Inter operators) of the framework are glued in two different ways (*LocalSearch* and *DoOperator)* with three different escaping mechanisms (SA,

ILS and VNS). The outline of the tested scripts is shown in the Appendix. Further development for reaching a better cumulative result of the Solomon benchmarks should focus on the methods for reducing the number of vehicles like ejection pool (Lim & Zhang, 2007) or ejection chain (Bräysy & Dullaert, 2003).

4.2 Characteristics of real-world problems

The four real-world VRPTW problems VRP1, VRP2, VRP3 and VRP4 are considered for optimization. Distribution of customers and vehicle routes are shown in Fig. 7. The set of standard VRP problems found in the literature and used to validate the performance of VRP solving algorithms use the Euclidian metric of distances. In contrast, solving the real-world VRP problems, due to the traffic rules and transport network topology requires the use of the traffic matrix. In the bidirectional traffic matrix the distances between the pairs of points stored in the transport layer of the Geographic Information System are not necessarily symmetric. This is most obvious for the routes in the urban areas while routes between urban areas are mostly symmetric. Solving the time constrained problems, as in the case of VRPTW, an additional matrix containing forecasted travel times data between each pair of customers has to be available.



a) VRP1 b) VRP2

c) VRP3 d) VRP4

Fig. 7. Maps showing customer's locations and vehicle routes

Therefore the problems are defined by two traffic matrices: the distance asymmetric look-up matrix and the related forecasted travel time matrix. The calculation of travel time matrix is based on the average velocity on a particular street or road segments. If such information is not available then the calculation is based on the rank of the road segments. In the example the road ranking follows the classification which divides them into sixteen categories. Customers for problems VRP1, VRP2 and VRP4 are located in the area of the city of Zagreb, the capital of Croatia and customers for VRP3 are located in a wider area of Zagreb. All described problems have heterogeneous fleet with two types of vehicles regarding different transport capacities (7 vehicles 2500 kg, 3 vehicles 3500 kg). The road networks are spread within big urban area, small cities and rural parts which gravitate to the capital and along inter-city highways. The number of customers varies for each problem (154, 234, 146 and 162). Overall loads per each problem are 11.1, 23.11, 14.8 and 14.6 tonnes of goods for delivery. Most of the customers have wide time widows from 7:00 a.m. till 2:00 p.m., except for a few customers who are located in the downtown area. The average customer service time is 10 minutes with little variation.

4.3 Real-world problems results

Table 2 shows which algorithm produces the best result for each of the real-world problems. Cost function was calculated by multiplication of a number of vehicles and overall distance. The quality thresholds *T1* and *T2* that are used in the comparative analysis are calculated by the procedure described in Section 3. Table 2 also shows average running time for each algorithm on each problem. That average time has important role in finding optimal number of iteration in performance measure protocol.

	ALG	*VEH*	*DIST*	*T1*	*T2*	*ARTA*		
						ILS	*SA*	*VNS*
VRP1	VNS	9	941117	8562195	8543250	4,42	4,43	2,97
VRP2	SA	10	1344551	14365330	14216120	13,68	14,16	8,87
VRP3	ILS	9	1287100	11659950	11626146	3,73	3,91	2,51
VRP4	ILS	9	847891	7711902	7655945	6,18	6,42	4,54

Table 2. Best results, thresholds and average running time of algorithms ILS, SA and VNS for real-world problems VRP1, VRP2, VRP3 and VRP 4. *ALG* = best performing algorithm, *VEH* = number of vehicles in solution, *DIST* = overall distance in meters, *T* = value of cost function, *ARTA* = average running time of algorithm in minutes

4.4 Comparative analysis of real-world problems results

The final results of the conducted experiments of performance measure protocol (see Section 3.) are shown in Table 3 and Table 4. The examination pool of results was constructed by 360 runs of the developed ILS, SA and VNS algorithms.

In order to determine which strategy needs less time, i.e. number of restarts multiplied by the number of iterations, to produce a solution below the threshold with some accuracy, each problem was solved 90 times with 30 runs of each algorithm.

Table 3 shows optimal parameters of the winning strategy for all problems. Parameters from Table 3 guarantee reaching of the threshold interval *T1* or *T2* in minimal time with 90% accuracy. For example, VRP1 needs to be restarted 4 times with halting criteria set to 89

iterations per start for ILS algorithm to reach threshold *T1* with 90% accuracy. If we increase the accuracy level the number of restarts increases.

Table 4 shows which algorithm, multi-start factor and halting number of iterations are optimal to reach threshold *T1* or *T2* with accuracy level 90%, 95% or 99.9% respectively and obtained by performance measure protocol. The thresholds are defined in such a way that all the results obtained by ILS, SA and VNS are sorted in a list where the value of the objective function on the last iteration is the number on which the sorting is done. Threshold *T1* is calculated so that 25% of the runs in the sorted list are in the *T1* threshold interval. Threshold *T2* has 10% of the best runs.

AL	**VRP1**			**VRP2**			**VRP3**			**VRP4**		
90%	***ALG***	***MSF***	***IT***	***ALG***	***MSF***	***IT***	***ALG***	***MSF***	***IT***	***ALG***	***MSF***	***IT***
T1	ILS	4	89	SA	6	99	ILS	68	5	VNS	34	6
T2	ILS	9	93	VNS	22	95	ILS	68	5	VNS	34	30

Table 3. Optimal tuning parameters for real-world benchmark problems VRP1, VRP2, VRP3 and VRP4. *AL* = accuracy level, *T* = threshold, *ALG* = algorithm, *MSF* = multi-start factor, *IT* = optimal number of iterations per each run

	VRP1						**VRP2**					
AL	***T1***			***T2***			***T1***			***T2***		
	ALG	***MSF***	***IT***	***ALG***	***MSF***	***IT***	***ALG***	***MSF***	***IT***	***ALG***	***MSF***	***IT***
90.0%	ILS	4	89	ILS	9	93	SA	6	99	VNS	22	95
95.0%	ILS	4	99	ILS	12	93	SA	10	75	SA	21	81
99.9%	ILS	10	89	ILS	26	93	SA	23	75	VNS	66	95

	VRP3						**VRP4**					
AL	***T1***			***T2***			***T1***			***T2***		
	ALG	***MSF***	***IT***	***ALG***	***MSF***	***IT***	***ALG***	***MSF***	***IT***	***ALG***	***MSF***	***IT***
90.0%	ILS	68	5	ILS	68	5	VNS	34	6	VNS	34	30
95.0%	ILS	89	5	ILS	89	5	VNS	44	6	VNS	44	30
99.9%	ILS	204	5	ILS	204	5	VNS	101	6	VNS	101	30

Table 4. Multi-start factors for real-world benchmark problems VRP1, VRP2, VRP3 and VRP4. *AL* = accuracy level, *T* = threshold, *ALG* = algorithm, *MSF* = multi-start factor, *IT* = optimal number of iterations per each run

The statistical analysis of 30 runs of each algorithm ILS, SA and VNS on each of the problems VRP1, VRP2, VRP3, VRP4 reveals that number of iterations which is 100 in the considered experiments is acceptable value for problems VRP3 and VRP4. In the case of other two problems VRP1 and VRP2 all optimal numbers of iterations are clearly grouped near the number 100 what leads us to the conclusion that the convergence of algorithms is not finished. The empirical study should be continued under the same protocol with the increased number of iterations for the problems VRP1 and VRP2. From the results in a Table 4, we can state that ILS is best performing algorithm for solving VRP3 which converges very early, but the number of restarts should be very large. With the large

numbers of restarts from 68 to 204 and only 5 iterations per run, the time duration to reach the threshold with ILS is more acceptable than time duration obtained by SA and VNS algorithms. The best performing algorithm for VRP4 problem is VNS algorithm.

5. Conclusion

For real-world application that solves the Vehicle Routing Problem it is essential to perform a fast selection of methods (constructive heuristics, neighbourhood operators and escaping mechanisms) which produce the desired improvement of the objective function. To speed up the prototyping a new flexible VRP framework has been developed and described.
The framework consists of the Framework Scripting Language (FSL), a library of coded methods and real-world benchmarks. Even new optimization ideas can use the advantages of this programming environment with tools considering commands that operate with VRP entities like moving vehicle, change customer position in routes, displaying the solving process of the current problem graphically, etc. The framework offers a set of programming tools to speed up the development, testing and tuning of heuristic algorithms.
The knowledge of solving practical problems by known methods is stored in the library which can be easily adapted for tailor-made application.
A new statistical approach for estimating the optimal number of restarts and iterations of the implemented algorithms is described and integrated in a general performance measure. This performance measure calculates for every solver the expected time that is necessary to compute solutions above a requested quality thresholds with respect to a demanded accuracy level.
The framework and the performance measure protocol are implemented on the standard benchmark and practical VRPTW problems.

6. References

Bräysy, O. & Dullaert, W. (2003). A Fast Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows. *International Journal on Artificial Intelligence Tools*, Vol. 12, No. 2, (June 2003) pp. 153-172, ISSN 0218-2130

Bräysy, O. & Gendreau, M. (2005a). Vehicle Routing Problem with Time Windows Part I: Route construction and local search algorithms. *Transportation Science*, Vol. 39, No. 1, (February 2005) pp. 104-118, ISSN 0041-1655

Bräysy, O. & Gendreau, M. (2005b). Vehicle Routing Problem with Time Windows Part II: Metaheuristics. *Transportation Science*, Vol. 39, No. 1, (February 2005) pp. 119-139, ISSN 0041-1655

Carić, T.; Fosin, J.; Galić, A.; Gold, H. & Reinholz, A. (2007). Empirical Analysis of Two Different Metaheuristics for Real-World Vehicle Routing Problems, In: *Hybrid Metaheuristics 2007*, Bartz-Beielstein, T. et al. (Eds.), Lecture Notes in Computer Science (LNCS) 4771, pp. 31-44, Springer-Verlag, ISBN 978-3-540-75513-5, Berlin/Heidelberg

Cerny, V. (1985). A Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm. *Journal of Optimization Theory and Applications*, Vol. 45, No. 1, (January 1985) pp. 41-51, ISSN 0022-3239

Clarke, G. & Wright, J.W. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points, *Operations Research*, Vol. 12, No. 4, (July-August 1974) pp. 568-581, ISSN: 0030-364X

Cordeau, J.-F.; Desaulniers, G.; Desrosiers, J.; Solomon, M. & Soumis, F. (2002). The Vehicle Routing Problem with Time Windows. In: *The Vehicle Routing Problem*, Toth, P. & Vigo, D. (Eds.), pp. 157-193, SIAM Publishing, ISBN 0-89871-498-2, Philadelphia

Flood, M.M. (1956). The Traveling Salesman Problem, *Operations Research*, Vol. 4, No. 1, (February 1956) pp. 61-75, ISSN: 0030-364X

Galić, A.; Carić, T. & Gold, H. (2006a). MARS - A Programming Language for Solving Vehicle Routing Problems. In: *Recent Advances in City Logistics*, Taniguchi, E. & Thompson, R. (Eds.), pp. 48-57, Elsevier, ISBN 0-08-044799-6, Amsterdam

Galić, A.; Carić, T.; Fosin, J.; Ćavar, I. & Gold, H. (2006b). Distributed Solving of the VRPTW with Coefficient Weighted Time Distance and Lambda Local Search Heuristics. *Proceedings of the 29th International Convention on Information-Communications Technology*, pp. 247-252, Opatija, May 2006, MIPRO, Rijeka, Croatia

Gillett, B.E. & Miller, L.R. (1974). A Heuristic Algorithm for the Vehicle-Dispatch Problem, *Operations Research*, Vol. 22, No. 2, (March-April 1974) pp. 340-349, ISSN: 0030-364X

Homberger, J. & Gehring, H. (2005). A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, Vol. 162, No. 1, (April 2005) pp. 220–238, ISSN 0377-2217

Hoos, H. & Stützle, T. (2005). *Stochastic Local Search: Foundation and Application*, Elsevier/ Morgan Kauffman, ISBN 1-55860-872-9658, San Francisco

Kirkpatrick, S.; Gelatt, C.D. & Vecchi Jr., M.P. (1983). Optimization by Simulated Annealing. *Science*, New Series, Vol. 220, No. 4598, (May 13, 1983) pp. 671-680

Laporte, G. (1992). The Vehicle Routing Problem: An Overview of Exact and Approximative Algorithms. *European Journal of Operational Research*, Vol. 59, No. 3, (June 1992) pp. 345-358, ISSN 0377-2217

Laporte, G. & Semet, F. (2002). Classical Heuristics for the Capacitated VRP, In: *The Vehicle Routing Problem*, Toth, Toth, P. & Vigo, D. (Eds.), pp. 109-128, SIAM Publishing, ISBN 0-89871-498-2, Philadelphia

Laurenço, H.R. & Serra, D. (2002). Adaptive search heuristics for the generalized assignment problem. *Mathware & Soft Computing*, Vol. 9, No. 2-3, pp. 209-234, ISSN 1134-5632

Le Bouthillier, A. & Crainic, T.G. (2005). Cooperative parallel method for vehicle routing problems with time windows. *Computers and Operations Research*, Vol. 32, No. 7, (July 2005) pp. 1685-1708, ISSN 0305-0548

Lim, A. & Zhang, X. (2007). A Two-Stage Heuristic with Ejection Pools and Generalized Ejection Chains for the Vehicle-Routing Problem with Time Windows, *INFORMS Journal on Computing*, Vol. 19, No. 3, (Summer 2007) pp. 443-457, ISSN 1091-9856

Mester, D.; Bräysy, O. & Dullaert, W. (2007). A multi-parametric evolution strategies algorithm for vehicle routing problems. *Expert Systems with Applications*, Vol. 32, No. 2, (February 2007) pp. 508-517, ISSN 0957-4174

Metropolis, N.; Rosenbluth, A.W.; Rosenbluth, M.N. & Teller, A.H. (1953). Equations of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, Vol. 21, No. 6, (June 1953) pp. 1087-1092, ISSN 0021-9606

Pisinger, D. & Röpke, S. (2005). A general heuristic for vehicle routing problems. *Technical Report*, Department of Computer Science, University of Copenhagen, Copenhagen, Denmark

Reinholz, A. (2003). Ein statistischer Test zur Leistungsbewertung von iterativen Variationsverfahren. *Technical Report 03027*, SFB559, University of Dortmund (in German)

Solomon, M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems with Time Windows Constraints. *Operations Research*, Vol. 35, No. 2, (March-April 1987) pp. 254-265, ISSN 0030-364X

Appendix – Pseudo code and main FSL scripts for SA, ILS and VNS algorithms

```
procedure SA()
   T := InitialTemperature()
   initial := InitialSolution()
   s := LocalSearch(initial)
   best := s
   while not Terminate() do
      s' := Perturbate(s)
      s" := LocalSearch(s')
      if f (s") < f (s) then
          s := s"
      else
          j := rnd(0, 1)
          k := -((f(s") - f(best)) / f(best)) /  T
          if j < exp(k) then
              s := s"
          endif
      endif
          if f (s) < f(best) then
              best := s
          endif
              T := CoolingSchedule()
    endwhile
return best
end
```

```
function SA(Solution initial) returns Solution
   Solution best, s, s1, s2
   boolean Terminate = false
   int MaxIterations = 100, counter = 0
   s = LocalSearch(initial)
   best = s
   double T = InitialTemperature(), j, k
   while not Terminate
        s1 = Perturbate(s)
        s2 = LocalSearch(s1)
        if better(s2, s) then
            s = s2
        else
            j = random()
            k = -((cost(s2) - cost(best)) / cost(best)) / T
            if j < exp(k) then
                s = s2
            endif
        endif
      if better(s, best) then
            best = s
      endif
      T = CoolingSchedule(T)
      counter = counter + 1
      if counter >= MaxIterations then
            Terminate = true
      endif
   endwhile
       return best
endfunction
```

```
procedure ILS()
   initial := InitialSolution()
   s := LocalSearch(initial)
   best := s
   while not Terminate() do
      s' := Perturbate(s)
      s'' := LocalSearch(s')
      if f (s'') < f (best) then
          best := s''
          s := best
      endif
   endwhile
   return best
end
```

```
function ILS(Solution initial) returns Solution
   Solution s, s1, s2, best
   boolean Terminate = false
   int MaxIterations = 100, counter = 0
   s = LocalSearch(initial)
   best = s
   while not Terminate
      s1 = Perturbate(s)
      s2 = LocalSearch(s1)
      if better(s2, best) then
          best = s2
          s = best
      endif
      counter = counter + 1
      if counter >= MaxIterations then
          Terminate = true
      endif
   endwhile
   return best
endfunction
```

```
procedure VNS()
   best := InitialSolution()
   op := 1
   while not Terminate() do
      success := DoOperator(op)
      if succes then
          op := 1
      else
          op := op + 1
      if op > OperatorCount() then
          if cost(CurrentSolution()) < cost(best) then
              DoIntraRoute()
              best := CurrentSolution()
          endif
              DoPreturbate()
              op := 1
      endif
      endif
   endwhile
   return best
end
```

```
function VNS(Solution initial) returns Solution
   boolean Terminate = false, success = false
   int MaxIterations = 100, counter = 0
   int operator = 1, operatorcount = 5
   Solution best = initial
   SetCurrentSolution(initial)
   while not Terminate
      success = DoOperator(operator)
      if success then
          operator = 1
      else
          operator = operator + 1
          if operator > operatorcount then
              if better (CurrentSolution(), best) then
                  intraRoute()
                  best = CurrentSolution()
              endif
                  counter = counter + 1
                  if counter >= MaxIterations then
                      Terminate = true
                  else
                      DoPreturbate()
                      operator = 1
                  endif
          endif
      endif
   endwhile
   return best
endfunction
```

3

An Effective Search Framework Combining Meta-Heuristics to Solve the Vehicle Routing Problems with Time Windows

Vincent Tam[1] and K.T. Ma[2]
[1]*Department of Electrical and Electronic Engineering, The University of Hong Kong*
[2]*DSO National Laboratories*
[1]*Hong Kong*
[2]*Singapore*

1. Introduction

Many delivery problems in real-world applications such as the newspaper delivery and courier services can be formulated as capacitated vehicle routing problems (VRPs) [10], which we want to route a number of vehicles with limited capacity in order to satisfy customer requests with the minimal operational cost. This is usually measured by the number of vehicles used multiplied by the total distance travelled. Often, a customer may specify a time-window with the earliest and latest time for the delivery, which gives rise to the VRP with time windows (VRP-TWs). In other words, a vehicle must arrive at a customer within the interval as specified by that customer in a VRP-TWs. The arrival of a vehicle before the earliest time as specified by a customer in a VRP-TWs will result in idle time. On the other hand, a vehicle is not allowed to reach a customer after the specified latest time. Moreover, a service time is commonly associated with servicing each customer. A real-life example of the VRP-TWs is furniture delivery in which each customer often requests the furniture to be delivery within a given period of a day. Unfortunately, the VRP-TWs are shown to be *NP*-complete, implying an exponential growth in the time complexity for a general algorithm to solve any of these delivery problems in the worst case. In practice, there are many instances of VRP-TWs involving 100 customers [1, 10] or more [9] which are difficult to solve optimally [1, 3, 6]. In the past two decades, VRP-TWs, due to their challenging nature and practical value, have continuously attracted many interesting proposals for useful heuristics and search algorithms [1, 3] to effectively solve problems in the areas of Artificial Intelligence [3], Constraint Programming [1] and Operations Research [3].

Among the heuristics [1, 2] proposed to solve the vehicle routing problems, there are some interesting proposals to initialize the search while others are targeted at advancing the search from a given initial state in an effective manner. As far as search initialization is concerned, there are two useful heuristics, namely the push-forward insertion heuristic (PFIH) [6] and the virtual vehicle heuristic (VVH) [2] which was proposed to generate more feasible initial states that may lead to better results. The PFIH is a simple-yet-efficient method to compute every route by comparing the cost of inserting a new customer into the

existing route against that of starting a new route in each iteration until all customers are served. However, when the capacity of a vehicle is exceeded or the delivery time must be dragged behind the latest time specified by the new customer, a new route has to be started. Clearly, the PFIH can only quickly return a feasible solution without any guarantee for its global optimality. On the other hand, the VVH works by using *virtual* vehicles with unlimited capacity to hold the deliveries that are not currently serviced by any real vehicle so as to allow a more optimized delivery plan to be computed. In other words, the virtual vehicles are used as temporary buffers to which no problem constraints (such as time and capacity) can be applied. Furthermore, to ensure all the deliveries will ultimately be performed by real vehicles, the cost incurred by a virtual vehicle for a customer visit is much higher than that incurred by a real vehicle. In this chapter, we will focus on comparing the influence of the above initialization heuristics on search meta-heuristics. Section 3 will present a more detailed discussion on the initialization heuristics.

After the initial routes for the VRPs are generated, we can apply many possible heuristic methods to improve on the current solution until a better delivery plan with lower operational cost is obtained. The Tabu search (TS) [1] is a well-known meta-heuristic possibly used for such improvement, which has also been successfully applied to solve many other combinatorial optimization problems [3]. In solving VRPs, given any initial route(s), there can be many possible *moves* [1] such as the 2-opt operation, which replaces any two links in a route with two different links to reduce the operational cost, to generate other possible route(s). The Tabu search works by firstly performing a neighborhood search on all possible moves, executing only non-Tabu moves which reduce the total operational cost, and 'memorizing' those recently performed moves with a Tabu list, usually of fixed length, to avoid cycling. In this way, TS promotes a diversified search from the current solution by continuously updating a short-term memory-like Tabu list until the predetermined stopping criterion is reached. In some cases, depending on the application-specific aspiration level criterion, the Tabu status of a move can be changed when it leads to a solution better than the current best. Alternatively, the guided local search (GLS) [2, 3] is another interesting meta-heuristic which has also achieved impressive results [2] in solving VRP-TWs effectively. Both TS and GLS are based on a local search operator to perform neighborhood search. However, each time after the local search is performed, the GLS uses a long-term memory-like penalty scheme to penalize all the undesirable *features* found in the current solution so as to avoid being trapped at the same local minima in future exploration. The resulting penalty information, after multiplied by a regularization parameter λ, is incorporated into an augmented objective function to *guide* the local search to iteratively look for a better solution from the current search position until a predefined stopping condition such as the maximum number of iterations is reached. In addition to solving VRPs, GLS has been successfully applied to solve many difficult scheduling problems such as the traveling salesman problem [12].

Previous work [1, 2] proposed to combine the possible advantages of different meta-heuristics for solving constrained optimization problems or in particular VRPs more efficiently or effectively. Wah *et al.* [4] considered the integration of simulated annealing technique in a Discrete Lagrangian search framework, which was closely related to GLS, as constrained simulated annealing (CSA) to solve a set of 10 constrained optimization problems (named G1 – G10) [4] with constraints and objective functions of various types. Later, the CSA was further improved with a genetic algorithm as CSAGA to achieve a better

efficiency, with a higher probability to obtain an optimal solution within a given period of time, for solving the same set of constrained optimization problems. Specifically, Genfreau *et al.* [13] integrated simulated annealing and Tabu search to effectively solve VRPs. Interestingly, Becker *et al.* [1], after comparing the individual performance of GLS and TS on the Solomon's test cases, tried to combine both TS and GLS as the guided Tabu search (GTS) method which outperformed the original meta-heuristics with an average of 1.7% better[1] in the solution quality on the "long haul" problems, that are problems involving deliveries of long distances. However, Becker *et al.*'s work suffered from three major drawbacks. First, their original aim, possibly only to obtain better experimental results, for combining both meta-heuristics was never explicitly stated. Second, the algorithm of the GTS was not clearly defined. There were only 3 statements to describe GTS very briefly. Third, no analytical model or clear explanation about the performance of GTS was given. On the other hand, we carefully propose in this chapter a different integration of GLS and TS as GLS-TS with a clear objective to combine the *best* possible advantages of applying both the short-term and long-term memory mechanism used in GLS and TS to solve VRP-TWs more effectively. More importantly, we provide a simple and easy-to-understand analytical model to understand the behavior of the resulting GLS-TS when compared to the original meta-heuristics. Furthermore, similar to Wah *et al.*'s work [4], we propose the integration of simulated annealing technique into the GLS-TS search framework as GLS-TS-SA. However, it is worth noting that simulated annealing is specially used as a *monitor* in GLS-TS-SA to carefully determine, depending on the relative merit of each meta-heuristic in the previous search, which meta-heuristic to apply in each iteration. To demonstrate the effectiveness of our proposals in solving VRP-TWs, we implemented the prototypes of both GLS-TS and its variant GLS-TS-SA. Both prototypes have been shown to compare very favorably in the solution quality against the original GLS and TS methods, and the search hybrid GTS on the well-known Solomon's test cases [10] which have been used as standard benchmarks for comparing the performance of different search strategies for decades. In addition, our proposed GLS-TS improved on one of the best published results, which were obtained from a number of advanced search techniques such as the Ant Colony search algorithm [9] and thus fairly difficult to break any of these records, in solving the Solomon's benchmarks on VRP-TWs. To our knowledge, this work represents the first attempt to systematically study the integration of TS into the GLS search framework.

This chapter is organized as follows. Section 2 describes the general vehicle routing problems and VRP-TWs in detail. The various search initialization heuristics such as the PFIH, interesting meta-heuristics such as the Tabu Search and combined meta-heuristics such as the Guided Tabu Search (GTS) to effectively solve VRP-TWs will be given in Section 3. Section 4 details our proposals of adapting the meta-heuristic search strategies to solve the VRP-TWs more effectively. In Section 5, we evaluate the performance of our proposal against the original heuristic search methods and the related GTS on the widely used Solomon test cases. Lastly, we conclude our work in Section 6.

---

[1] It should be noted that Solomon's test cases [10] have been tackled by researchers for decades, thus leaving very little room for improvement. Besides, the original TS [1] and GLS [3] methods already achieved relatively good results as compared to the best published results. Thus, the average improvement of 1.7% on "long haul" problems is comparatively significant.

2. The vehicle routing problems with time windows

A general vehicle routing problem (VRP) [1, 10], which is shown to be *NP*-complete [3], can be formally defined as follows. We are given a fixed N or infinite number of vehicles with limited capacity cv (measured in weight or volume) and M customers' requests, in which each request rq_j demands a delivery service for Q_j quantity of goods/service, to different locations. The distance, usually measured in term of minutes or hours required for travel, between any two possible delivery points is also provided, usually as a distance matrix. Then, our task is to optimize certain user-defined criteria subject to the following basic constraints:

1. any customer request rq_j should only be served by one vehicle only;
2. for each vehicle v, the sum Qv of quantity of goods to be delivered by vehicle v must be less than or equal to cv;

Besides the above basic constraints, in many real-life applications such as supermarket delivery, each customer may request the items to be delivered in a given period (time window) of a day. Thus, a vehicle routing problem with time windows (VRP-TW) has an additional time-window constraint to be imposed for each delivery as follows.

3. when a time window with the earliest time E_j and the latest time L_j is specified for each delivery in a VRP-TW, the arrival time T_{vj} of vehicle v to serve customer request rq_j must lie within the specified duration, that is $E_j \le T_{vj} \le L_j$.

One of the most common objectives for minimization in VRPs or VRP-TWs is $TV \times TD$, where TV is the number of vehicles used, and TD is the total distance travelled by all vehicles. Throughout this chapter, our compared search strategies work to minimize this common objective of $TV \times TD$. Clearly, there can be many variants of VRPs or VRP-TWs with different objectives such as the total travel time of all the vehicles or the total waiting time of all the customers for optimization in many real-life applications. In the next section, we examine two common heuristics, namely the push-forward insertion heuristic and the virtual vehicle heuristic, useful to obtain an initial and feasible solution in solving the VRPs or VRP-TWs. Moreover, we will review two interesting meta-heuristics to solve the difficult VRP-TWs optimally.

3. Useful heuristics and meta-heuristics for vehicle routing

Since the search space for all the possible (feasible or slightly infeasible) routes in VRPs or VRP-TWs can be fairly large even for instances involving 100 customers [1, 10] or more [9], and the time-window constraints in the VRP-TWs can be difficult to satisfy, the careful choice of a suitable heuristic to return only feasible, and possibly more optimal, solutions can be important for further optimization. The push-forward insertion heuristic [6] and virtual vehicle heuristic [2] are two useful heuristics for search initialization in solving difficult VRPs. In addition, we will examine in this section two well-known meta-heuristics, namely the guided local search (GLS) and Tabu search (TS), which are based on completely different memory-like control mechanisms to restrict the local search operator in continuously optimizing the current solution, after input from the heuristic initialization method, until an optimal solution is obtained to solve the VRP-TWs successfully. TS uses a short-term memory-like Tabu list to avoid cycles in search. On the other hand, GLS uses a long-term memory-like penalty scheme to "memorize" all the undesirable features occurring in the previously visited local minima. In Section 4, we will examine various proposals to integrate TS into the GLS framework to solve VRP-TWs more effectively.

3.1 Useful search initialization heuristics

The push-forward insertion heuristic (PFIH) [10], introduced by Solomon in 1987, is an efficient method to compute a feasible solution for any VRP by assuming an infinite number of vehicles. The PFIH starts a new route by selecting an initial customer, usually farthest from the delivery depot, and then iteratively inserts any unassigned customer into the current route until the capacity of the current vehicle is exceeded or the waiting time for any newly added customer will exceed its associated duration constraint. At this moment, a new route will be created. And this process repeats until all customers are served. Basically, the algorithm for PFIH can be summarized as follows.

1. Begin an empty route r_0 starting from the depot; *i := 0;*
2. Among all unassigned customers, select the customer farthest from the depot (in case there is a tie, break the tie randomly) and insert into the current route r_i;
3. **If** all customers are routed, **then goto** step 6. **else** :
   If the capacity *cv* of the vehicle *v* involved in the current route r_i is exceeded, **then goto** step 5. **else** :
   foreach (unassigned customer)
   find the best position for insertion in r_i to
   compare the cost of starting a new route against that for the best position found.
4. Pick the customer with the greatest cost difference and insert it into r_i. Update the capacity *cv* of the vehicle *v* involved. **goto** step 3.
5. Start a new route r_{i+1} starting from the depot; *i := i+1;* **goto** step 2.
6. Return the current solution.

Figure 1. The Algorithm for Push-Forward Insertion Heuristic (PFIH) to Solve VRPs

Clearly when handling VRP-TWs, we have to adapt the last **else-**part of step 3 as below.

foreach (unassigned customer)
find the best position for insertion in ri **without violating any specified time-window** to compare the cost of starting a new route against that for the best position found.

Moreover, we have to add the following conditional statement to the beginning of step 4 as follows.

If there exists no feasible position for insertion into the current route r_i, **then goto** step 5. **else** :

In general, PFIH can be integrated into the search framework of many heuristic search algorithms [3, 6] to efficiently handle the VRP-TWs. It should be noted that the initial solution returned by PFIH represents only a feasible and usually non-optimal solution for further optimization by the different meta-heuristics.

On the other hand, the virtual vehicle heuristic (VVH) [2], with the availability of application-specific knowledge such as the number of vehicles used in the best published result for a particular VRP-TW, can often return both feasible and fairly optimal initial solutions for further processing. Using this approach, a virtual vehicle is used to hold the unassigned customers. The virtual vehicle is different from a real vehicle in three respects. First, the virtual vehicle cannot visit two or more customers on the way, but it has to visit each customer in turn after making a return trip to the depot. Secondly, it is not constrained

by any domains. Third, the cost of a customer visit for a virtual vehicle is higher than a real vehicle. This is to ensure that all customers will ultimately be assigned to a real vehicle. The VVH method for search initialization can be specified formally as follows.

Given

- an objective function O (often measured in term of the total distance travelled by all vehicles)
- the cost of delivery by a virtual vehicle = $\alpha_1 (D_{di} + D_{id}) + \alpha_2$, where
  - D_{ij} is the distance between customers c_i and c_j with d as the depot,
  - α_1 and α_2 are parameters set to increase the cost of a visit by the virtual vehicle,

the VVH algorithm proceeds as follow.

1. The virtual vehicle services all the customers' requests.
2. For each customer-vehicle combination in the current solution, use 4 heuristic operators, namely the *2-opt*, *relocate*, *exchange* and *cross*, to try to improve the solution quality by firstly considering only legal moves that do not violate any constraint, and then executing the legal move which decreases the objective function most.
3. **If** no more customer-vehicle combination can be improved to reduce the solution quality, **goto** step 5.
4. **goto** step 2.
5. **return** the current solution.

The four heuristic operators used in step 2 try to improve the current solution by changing the order to serve different customers in the same or different routes. These operators can be classified as intra-route or inter-route. Figure 2 shows the 4 heuristic operators.

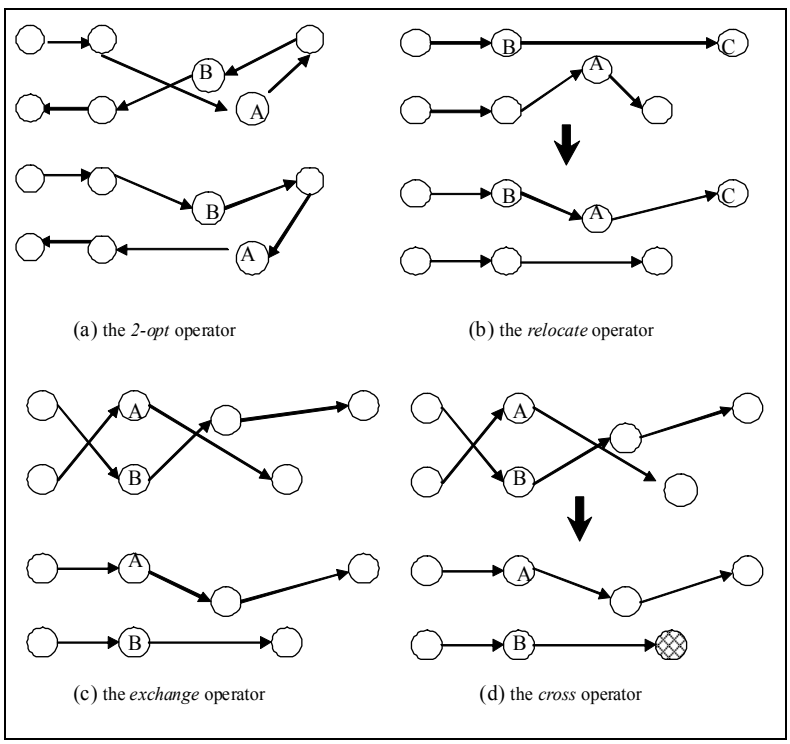


Figure 2. The 4 Heuristic Operators for Improving Routes in Solving VRPs

The *2-opt* [14] is an intra-route operator which reverses a section of a route by deleting 2 arcs, and replacing them with any 2 arcs to reform the route. On the other hand, the *relocate, exchange* and *cross* [15] are inter-route operators. The *relocate* operator moves a visit from its position in one route to another position in either the same or a different route. *Exchange* swaps 2 visits from either the same or different route while *cross* swaps the end portions of 2 routes.

Besides, from the empirical results obtained in [1], good settings of parameters to calculate the cost of customer visits performed by virtual vehicles are $\alpha_1 = 1.025$ and $\alpha_2 = 0.0005\Delta$, where Δ is the distance between the two most remote sites in the problem. Also, to achieve reasonable results in solving the VRP with VVH, we need to set the maximum number of vehicles allowed. This observation is consistent with previous work [2] in which the maximum number of vehicles allowed for VVH should be v_p, v_p+1 or v_p+2, where v_p is the number of vehicles used for the best published results. Nevertheless, the stringent demand for this extra parameter v_p to improve the quality of the initial solution produced will make VVH a more application-specific heuristic, thus less favorable for any general real-life application.

3.2 Guided local search

Guided local search (GLS) uses a meta-heuristic based on penalties to continuously modify the penalty terms associated with an augmented objective function to escape from local minima for efficiently solving a wide range of constrained optimization problems [3]. Based on the original objective function *O(S)* for optimization, GLS defines an augmented objective function as: $O'(S) = O(S) + \lambda \sum_{i \in F} f_i(S).p_i.c_i$ where λ is a penalty factor representing the relative importance of all the penalties, $f_i(S)$ is an indicator function returning *1* when the feature *i* in the set *F* of features under consideration appears in the current solution *S*, and *0* otherwise; p_i is the number of times which the feature *i* is penalized, and c_i represents the cost of feature *i*. With all these input parameters properly set for a specific application, the GLS algorithm [2, 3] works as in Figure 3.

The function *InitialSolution()* returns the initial solution to solve the VRPs by PFIH or VVH as previously described. *LocalSearch(S)* performs a local search on the current solution *S* till no more improvements can be made. The local search is executed based on the 4 heuristics operators described in the previous section. Similar to the VVH, the "best accept" approach is used in which the move which reduces the value returned by the objective function *O* by the largest amount will be executed. *StoppingCondition()* checks whether the predetermined stopping criterion to terminate the search is satisfied in each iteration. In many real-life applications, the stopping criterion can be defined in terms of the maximum number of iterations or improving moves. *Choose Penalty Features (S, p)* takes the current solution *S* and the penalty vector *p*, and then returns the set of features f_i to be penalised. For general applications, GLS chooses all features in S for which the utility $c_i/(p_i+1)$ is largest amongst the features in S. The statements marked with (1) and (2) are important for the subsequent discussion of search hybrids based on GLS.

```
P := 0⃗ ;                    // initialise the penalty vector to be zeros
S := InitialSolution();   // S is the current solution
S* := LocalSearch(S);   //Make sure the best solution S* found starts from a local minimum
while not StoppingCondition() do
  f := ChoosePenaltyFeatures(S, p);----------(1)
  foreach x in f do p_x := p_x + 1; ----------(2)
  S := LocalSearch(S);
  If O(S) < O(S*) then S*:= S;
return S*
```

Figure 3. The Pseudo-codes of Guided Local Search

It is interesting to note that the penalty vector *p* serves as a long-term memory to memorize all the undesirable features appearing in some previously visited local minima. In solving VRPs, GLS or its related search method selects the longest arc (that is the arc of the greatest distance between any two nodes in a route) as an undesirable feature to be penalized for the current solution, weighted by the number of times that longest arc has been already penalized. Furthermore, there were some empirical observations [2, 3] stating that the value of λ might greatly affect the performance of GLS and the quality of the solution returned. As for solving VRP-TWs, it was discovered in [2] that GLS worked well when λ was ranged from *0.1* to *0.3*, giving the best result at *0.2*.

3.3 Tabu search

Tabu search is an interesting meta-heuristic which aims to model the human memory process through the use of a Tabu list, usually of a fixed length, to prohibit most recent *moves* possibly leading to some previously visited local optima. In solving VRP-TWs [1, 10], moves can be defined in term of the addition or deletion of arcs associated with individual nodes (as customers) into or out of all computed routes. Besides the Tabu list, an *aspiration* criterion is used to change the Tabu status of a move so that the move in the Tabu list can still be accepted when it leads to a solution better than the current best solution. In general, the performance of a Tabu search algorithm depends largely on effectiveness of the local search operators, the length of the Tabu list and the settings used for the aspiration criterion. Nevertheless, Tabu search has been extensively used to solve many difficult combinatorial problems with good results [1].

Figure 4 gives the Tabu search algorithm as described in [1, 2].

The search starts with an initial solution *S* provided by *InitialSolution()*. Then, *LocalSearch(S)* uses the four heuristic intra-route and inter-route operators to perform a local search so as to iteratively improve the current solution *S* until it reaches a local minimum which is then assigned to the current best solution *S**. Before a predefined stopping condition is met, *RankMove(S)* firstly returns a ranked list of all possible moves, ordered by decreasing cost difference, from the current solution *S*. While there is no move performed, the first non-Tabu move *m*, as determined by *IsNotTabu(m)*, from the list *moves* will be executed by *Perform(m)*. Then, the flag *moved* will be reset to exit the inner loop before *InsertTabuList(m)* adds the most recently performed move *m* into the Tabu list. Lastly, the current best solution found may be updated, for which *O(S)* is the objective function returning the quality of the current solution *S*. The whole process is repeated until the stopping criterion is reached. For

details, refer to [1]. It should be noted that, the above discussion mainly focuses on the use of Tabu list as a simple short-term memory to diversify the search for avoiding local optima. However, it may also be possible to include some long-term memory structure for explicit intensification and diversification phrases as in some sophisticated Tabu search algorithms. In fact, in the guided Tabu search algorithm (GTS) [1] or our proposal of integration to be discussed in Section 4, the penalty-based learning scheme [2, 3] can also be regarded as one possible type of long-term memory mechanism for avoiding the already visited local optima. The statement marked with (*) is used for the subsequent discussion of GTS.

```
S := InitialSolution();   // S is the current solution
S := LocalSearch(S);      // Make sure we start with a local minimum
S*:= S;                   // S* represents the best solution found
while not StoppingCondition() do
  moves := RankMoves(S);  ----------(*)
  moved := false;
  while not moved do
    m := head(moves);
    if IsNotTabu(m) then
      Perform(m);
      moved := true;
     InsertTabuList(m);
    If O(S) < O(S*) then S*:= S;
return S*
```

Figure 4. The Pseudo-codes of Tabu Search

4. Combining meta-heuristics to effectively solve VRP-TWs

As discussed in the previous section, Tabu search employs a Tabu list as the short-term memory to avoid cycles whereas the guided local search (GLS) uses a penalty vector as the long-term memory to memorize all undesirable features occurred in the previously visited local minima. Therefore, it can be advantageous to combine both meta-heuristics to complement each other during the search process. In the following, we discuss the various proposals for combining both TS and GLS to solve VRP-TWs effectively. First, we review the Guided Tabu search (GTS) as an ad hoc integration of TS and GLS proposed by Backer *et al.* [1]. Secondly, we consider our careful proposal of combining TS and GLS as the GLS-TS algorithm with an easy-to-understand and interesting analytical model to differentiate its expected search behavior from that of GTS. Lastly, we will describe an integration of simulated annealing technique into the GLS-TS framework as a variant of GLS-TS to demonstrate the numerous opportunities opened up by our proposal of integration.

4.1 Guided Tabu Search

Originally, Backer *et al.* [1] aimed at comparing the performance of GLS and TS in solving the Solomon's test cases of VRP-TWs. Later in the experimentation stage, after observing the simplicity and impressive results produced by GLS, Backer *et al.* proposed a fairly ad hoc integration of the GLS penalty scheme into the Tabu search algorithm as the Guided Tabu

search (GTS) to possibly improve on the performance of the original search methods. It should be noted that GTS was only briefly mentioned in [1] without showing any explicit pseudo-code. To facilitate our subsequent discussion, we explicitly define the GTS algorithm as follows.

```
S := InitialSolution();   // S is the current solution
S := LocalSearch(S);   // Make sure we start with a local minimum
S*:= S;              // S* represents the best solution found
while not StoppingCondition() do
 f := ChoosePenaltyFeatures(S, p); // to select and then penalize the concerned features
 foreach x in f do p_x := p_x + 1;      // so as to avoid re-visiting the same local minima
 moves := RankMoves(S);
 moved := false;
 while not moved do
  m := head(moves);
  if IsNotTabu(m) then
   Perform(m);
   moved := true;
  InsertTabuList(m);
  If O(S) < O(S*) then S*:= S;
return S*
```

Figure 5. The Pseudo-codes of Guided Tabu Search

Figure 5 gives the pseudo-codes of the GTS algorithm. Syntactically, the GTS algorithm is obviously obtained by inserting lines (1) and (2) of the GLS algorithm as shown in Figure 3 before the line (*) of the TS algorithm as described in Figure 4. The semantic meaning of this addition is revealed in Backer *et al.*'s original work [1] which clearly stated that after each move, a GLS-type procedure was called to update the weight in the cost matrix (used in the penalty scheme). However, we consider this frequent invocation of the GLS-type procedure may collect "immature and hazardous" penalty information after every single *diversifying move* of TS. The collected penalty information can be "hazardous" in two senses. First, it can mislead the current search direction by biasing towards some unimportant features found in the previous solutions. Second, the vast amount of possibly useless penalty information generated may simply cause information or memory overflow during program execution. Besides, it may slow down the performance of GTS. In the next subsection, we will provide an interesting and simple model to carefully consider the possible influence of this frequent penalty mechanism on the search behavior of GTS. After all, from Backer et al.'s empirical experience [1], GTS showed some benefits by outperforming the TS and GLS with an average of *1.7%* on long-haul problems, including C2, R2 and RC2, of the Solomon's test cases. However, for the remaining problem classes, GTS did not show any convincing result when compared to GLS.

4.2 Our proposed Guided Local Search - Tabu Search (GLS-TS)

As previously discussed, the guided Tabu search (GTS) did not seem to be an attractive proposal for integrating GLS and TS since the short-term memory nature of the TS may *not*

be able to effectively use the long-term memory-like penalty information *frequently* provided by the GLS scheme to diversify the search from its current status. In other words, the GLS scheme may penalize some features which will be totally irrelevant to the current state of TS. Therefore, we propose an alternative integration: instead of performing a single non-Tabu move for every update of the cost matrix used in the penalty scheme, we propose to update the penalties only when the TS reaches a *stable state*. In other words, the penalties are updated only when all the non-Tabu moves cannot further improve the current solution any more.

Figure 6 shows the GLS-TS algorithm. The definitions for the function InitialSolution(), LocalSearch(), StoppingCondition(), RankMoves(S), head(moves), IsNotTabu(m), InsertTabuList(m) and ChoosePenaltyFeatures(S,p) follow those previously described for the GLS and TS algorithms. The Boolean variable improved is used to detect whether the TS has reached a stable state when no improvement on the current solution can be made by any non-Tabu move. Although our proposed GLS-TS may be seen carelessly as a slight modification from the GTS, the implication behind such modification can be significant. By appropriately integrating the penalty information of GLS only when the Tabu search is stable, our proposed GLS-TS algorithm tries to maximize the best advantages of information gained from GLS to improve the original Tabu search. This competitive advantage of GLS-TS when compared to GTS is illustrated clearly in the following simplified Markov chain diagrams to analyse their different search behaviour in successfully finding a solution for general and solvable VRPs.

```
P := 0⃗ ;                    // initialise the penalty vector to be zeros
S := InitialSolution(); // S is the current solution
S* := LocalSearch(S); // Make sure the best solution S* found starts from a local minimum
while not StoppingCondition() do
 f := ChoosePenaltyFeatures(S, p);
 foreach x in f do p_x := p_x + 1;
 improved := true;    ----------------------------------------------(**)
 while improved do // Start of TS Local Search
       prevO := O(S);
       moves := RankMoves(S);
       moved := false;
       while not moved do
            m := head(moves);
             if IsNotTabu(m) then
                Perform(m);
                moved := true;
                InsertTabuList(m);
        if O(S) < O(S*) then
                  S*:=S;
        If prevO <= O(S) then
                  improved := false; // End of TS Local Search ----------(***)
 return S*
```

Figure 6. The Pseudo-codes of Guided Local Search – Tabu Search (GLS-TS)

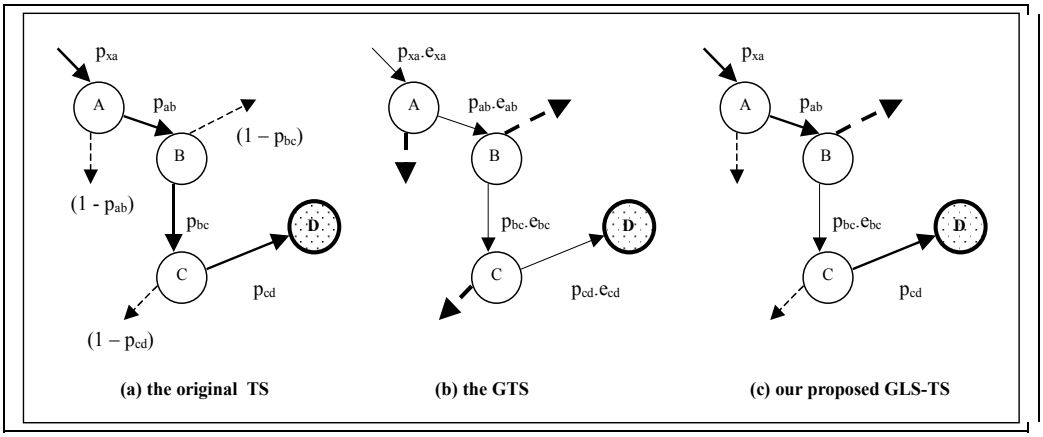


Figure 7. The simplified Markov models for the original TS, GTS and GLS-TS algorithms

Figure 7 gives the simplified Markov models for the original TS, GTS and GLS-TS algorithms which successfully find a solution for a general and solvable VRP. Each circle in the diagram is a (search) state representing the recently computed routes for all the involved vehicles to solve a VRP. A solution state is a terminal state for which a solution is found for that particular VRP. Each arrow, attached with the corresponding probability, represents a state transition from one state to another possible state after applying any of the heuristic operators as described in Figure 2. For simplicity, we assume for a specific VRP with only one solution, the last four states: A, B, C and D are identified, out of a careful empirical observation on all successful runs, with state B as the last 'stable' state representing a local minimum before reaching the sole solution state D. Figure 6(a) shows the state transitions with their attached probabilities for these last four states of the original TS as the basis for subsequent comparisons against the GTS and our proposed GLS-TS. Due to space limitation, an arrow with a solid line from the current state readily denotes a transition from the current state to the targeted state *X* so as to ultimately reach the solution state with the minimal number of transitions whereas an arrow with a dotted line is used to represent a transition from the current state to any non-*X* state. The thickness of the (solid- or dotted-) line of any arrow is proportional to the likeliness/probability for that transition to occur. For instance, the solid-line arrow from state A to state B in Figure 6(a) is attached with the probability p_{ab} for this transition to occur while the dotted-line arrow from state A to any non-B state is attached with the probability $(1 - p_{ab})$. Presumably, the probability p_{ab} is higher than $(1 - p_{ab})$ since the solid-line is clearly thicker than the corresponding dotted-line. In general, we assume the recurring property for the above simplified Markov models, that is a non-zero probability guaranteed for any state (including any non-*X* state) to reach any other possible state (such as the solution state) in the model, the sum of probabilities for all possible transitions out of any particular state is always 1, and lastly, to compute the probability for a sequence of state transitions, each transition is taken as an independent event according to the theory of *Probability* [16]. Therefore, given the recurring property of the simplified Markov model, the probability of reaching the solution state D from any non-*X* state is always $1/ (N - 1)$ where N is the total number of possible states in the simplified model. Hence, the probability of reaching from state A successfully to state D equals to $p_{ab} . p_{bc} . p_{cd} + (1 - p_{ab}). 1/ (N - 1) + p_{ab} .(1- p_{bc}). 1/ (N - 1)$, in which and the second term $(1 - p_{ab}). 1/$

(N - 1) gives the probability of reaching from state A to non-B state and finally to state D. Furthermore, we assume the probability of reaching from any previous state to state A is greater than 0.5 so that the probability of successfully reaching state D largely depends on the decisions made in the last four states. Since the three probabilities p_{ab} , p_{bc} and p_{cd} normally range from 0 to 1, and N is at least of the order of *1,000* or more, the first term $p_{ab} . p_{bc.} p_{cd}$ is usually the most important quantity to consider in determining the probability of reaching from state A to state D. To illustrate, let p_{ab} , p_{bc} and p_{cd} be *0.2*, *0.3* and *0.4* respectively, and N be *1,000,* the probability of reaching from state A to state D equals to $0.2 \times 0.3 \times 0.4 + (1 - 0.2) \times 1/999 + 0.2 \times (1 - 0.3) \times 1/999 \cong 0.024 + 0.0008 + 0.00014 \cong 0.02494$ in which the first term is clearly the most determining factor. However in the GTS, by including the GLS penalty mechanism in every step of the TS, we assume the probability of a transition (A $\rightarrow$ B) being affected by the GLS penalties as g_{ab}, and conversely the probability of not being affected as $e_{ab} = 1 - g_{ab}$. Similarly, all of these *e's* and *g's* will be in the range of 0 to 1. Moreover, the *g's* should be progressively increasing from the initial to the last state since there can be more features to be penalized or higher penalty resulted after each iteration which may affect the state transition of the original TS more drastically. Overall speaking, applying the GLS penalty scheme in each TS step will substantially lower the first important term of the probability of successfully reaching from state A to state D, at least by the order of 10^{-3}, to $p_{ab} . p_{bc.} p_{cd.} e_{ab} . e_{bc.} e_{cd}$ which cannot be offset by the relatively small opposite increase in the second and third terms of the specific probability due to the very small increase in the probability of reaching the non-B and non-C states from state A and B. This clearly shows the possible pitfalls of the GTS proposal which may simply increase the chance of diverting the current search to some unintentional states, that are any states other than the solution state D in the non-B and non-C states. On the other hand, by intelligently maintaining the original characteristics of the TS and only affect the TS by the GLS penalty mechanism in *stable states* like state B in our proposed GLS-TS as shown Figure 6(c), the original strength as reflected in the first important term of the probability of reaching from state A to D is largely unaffected by multiplying only one e_{bc} as $p_{ab} . p_{bc.} e_{bc.} p_{cd.}$. Meanwhile, the probability of reaching non-C states at the stable state B is appropriately increased to open up a *risky but possibly favorable* opportunity to try to directly reach the solution state D. Also, it is worth noting that since the system already arrives at the stable state B, there is a fair chance of re-entering the same stable state B even after it fails to reach the solution state D. In other words, we carefully inject some intended *noise* into the original search model of TS (as shown in Figure 6(a)) to increase the chance of directly reaching the solution state(s) only at stable states while not drastically scarifying the original strength, that is a higher probability of following the normal transitions from A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ D, of TS in our proposed GLS-TS. After all, Section 5 will give the experimental results of the original TS, the GTS and our proposed GLS-TS on the well-known Solomon's benchmarks [10] to justify about our discussion here.

4.3 Another variant of GLS-TS

Both GTS and our proposed GLS-TS are basically *fixed* strategies to invoke the GLS-type penalty scheme within the TS framework, it will be interesting to have other variants of GLS-TS which can flexibly invoke different search mechanisms depending on the relative merit of the involved mechanisms during the dynamic search process. Basically, we need a kind of measure and a *monitor* as the feedback-and-control mechanism of this flexible

invocation scheme. Similar to Wah *et al.*'s work [4], we propose to integrate simulated annealing as a monitor to supervise the performance of different search mechanisms and decide which mechanism to invoke in each stage of the search. This forms the basis of our proposed GLS-TS-SA algorithm as a variant of GLS-TS. This will decide whether to invoke the common local search method used in GLS or the unique Tabu search method during the search process. For testing, we simply used the number of accumulated *improving moves* made by each search method as a score to measure their relative merit. Obviously, there can be more sophisticated measures defined to evaluate their performance. More importantly, our proposal of integration discussed here opens up many potentially interesting directions for future investigation.

```
P := 0⃗ ;                    // initialise the penalty vector to be zeros
S := InitialSolution(); // S is the current solution
S* := LocalSearch(S); // Make sure the best solution S* found starts from a local minimum
i:=0                        // initialise i
impM:=0                     // initialise the counter impM
while not StoppingCondition() do
  f := ChoosePenaltyFeatures(S, p);
  foreach x in f do p_x := p_x + 1;
  temp:= MAX_TEMP * e -(impM)/i
  P_calc := 1 / (1 + e(-d/temp))
  P_gen := Random(0,1)
  if P_gen <= P_calc then
    S:=Tabu-LocalSearch(S)
    i:=i+1
    if O(S) < O(S*) then
        impM:=impM+1
   else
     S:=LocalSearch(S)
  if O(S) < O(S*) then
            S*:=S;
return S*
```

Figure 8. The Pseudo-codes for the GLS–Tabu Search integrated with Simulated Annealing (GLS-TS-SA)

Figure 8 gives the pseudo-codes of the GLS-TA integrated with simulated annealing. *InitialSolution()*, *LocalSearch()*, *StoppingCondition()* and *ChoosePenaltyFeatures(S,p)* are as described in the GLS-TS algorithm in Figure 6. *Random(0,1)* generates a random real number between *0* and *1*. The function *Tabu-LocalSearch()* basically performs a Tabu search until no more improvements for the objective function is achieved. The variable *i* counts number of times the variant GLS-TS-SA performs a Tabu search while the counter *impM* measures the number of accumulative improving moves affected by Tabu search. The variable *d* represents the *decrease* in the total distance traveled, which is either *0* or *-d* for any increase. The parameter *MAX_TEMP* was empirically determined at 0.3. In addition, the formulation for *temp* and P_{calc} basically follows the standard logistic function described in [11]. Besides our proposed GLS-TS-SA, we can clearly have many other possible search hybrids

integrated with the simulated annealing such as an adaptive strategy which can flexibly invoke the GLS penalty scheme within the TS framework depending on its performance. Some of these possible extensions will be discussed in Section 6.

5. Experimental results

We used Solomon's test cases to evaluate the performance of our proposals against the original Tabu search [13] and Backer's *et al.*'s variant - the Guided Tabu search [1]. There are *56* instances of delivery problems in the Solomon's test set, each with 100 customers, which can be categorized into six classes: C1, C2, R1, R2, RC1 and RC2. The first letter(s) of the class name denotes the type of customer distribution. For instance, 'C' represents the clustered customers, 'R' denotes the randomly distributed customers, and 'RC' involves a mix of customers of both types. After that, the number in the class name encodes the types of vehicle capacity and service duration. '1' refers to the test cases with small vehicle capacity and short service duration while '2' stands for those with large vehicle capacity and long service duration. For all the six classes, the demand for each customer within the same class is the same.

The original guided local search (GLS) and Tabu search algorithms and their hybrids such as the Guided Tabu search (GTS), our proposed guided local search – Tabu search (GLS-TS) and its variant GLS-TS integrated with simulated annealing (GLS-TS-SA) are all implemented in C/C++ and compiled with g++ (the GNU C++ version egcs-*2.91.66* with optimised compilation option), running on a machine with Intel Pentium III (450Mhz) processor and *256* megabytes of random-access memory (RAM). The operating system is Linux RedHat Version *6.1* (Cartman) Kernel *2.2.12-20* on i686. All the implementations are in fact modified from the original PFIH+MGA [6]. Most of the data structures are implemented as global arrays for efficient handling. Moreover, the four heuristic operators are modified so that instead of performing an actual move, the effects of performing the move are calculated. This is achieved by considering the arcs added and deleted for a particular move. For the constraint checking, the new move is only checked against those routes it will affect. Hence, the execution time of the program is greatly reduced by a factor of *10* or more.

The lemma values used are *0.2* for GLS, and 0.15 for GTS, GLS-TS and GLS-TS-SA. For each approach, the solver is executed *10* times with the average and the best results recorded. For all the approaches, the resource limit or stopping criterion is set at *1000* iterations. For the following results, we use "PFIH+" to represent those search methods using the push-forward insertion heuristic for initialisation while "VVH+" is used to represent those methods using the virtual vehicle heuristic as discussed in Section 3.

Table 1 summarizes the performance of different solvers initialised with PFIH or VVH in solving the Solomon's test set [10] of VRP-TWs. The performance of the solvers is measured in terms of two quantities: the sum of $TV \times TD$ and the averaged deviations from the best published results from six sources [17] in percentage for all the test cases. From these experimental results, GLS-TS outperforms against both GTS regardless of the initial search. For instance, GLS-TS remains competitive when comparing the PFIH+GLS with PFIH+GLS-TS. Furthermore, in term of the averaged deviations from the best results, VVH+GLS-TS with the lowest *12.26%* is the winner among the GLS- or TS-based solvers we compare here.

It is worthwhile to note that VVH+GLS-TS has improved on 1 test case, R205, giving: *TV=997.385, TD=3* and *TV* × *TD=2992.16* which is *0.13%*[2] better than the best result: *TV=998.72, TD=3* and *TV* × *TD=2996.16* as reported in [2]. In general, it can be very difficult to break any of these best-published records since they are in fact the best results among the best solvers ever compared. This supports our analysis discussed in Section 4.2 that updating the penalty vector only at local minima rather than after each Tabu move may significantly improve the performance of the original search methods. These results demonstrate that our proposal, GLS-TS, has competitive advantages over its parents GLS and TS in solving VRP-TWs effectively.

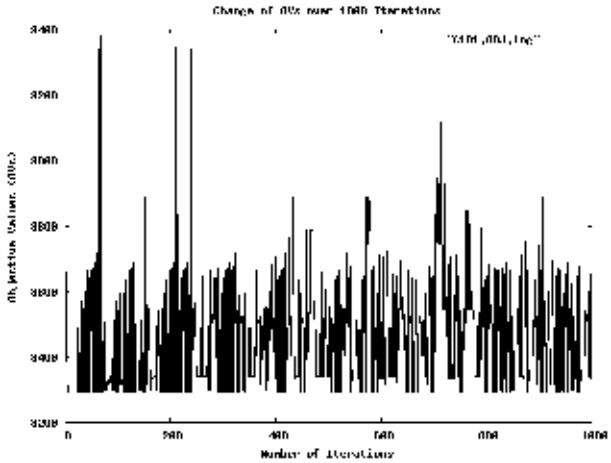
Approaches	**TVxTD**	**Averaged differences of all test case from BEST (%)**
BEST	453997.1	0
PFIH+GLS	503206.1	14.21
PFIH+GTS	507321.2	15.59
PFIH+GLS-TS	504924.3	14.53
PFIH+GLS-TS-SA	556400.3	16.10
VVH+GLS *	486590.1	14.83
VVH+GTS*	455471.0	14.74
VVH+GLS-TS*	480169.8	12.26
VVH+GLS-TS-SA	497175.1	12.96

Table 1. A Comparison of the Performance of All the Solvers initialised with the PFIH or VVH Against the Best Results on Solomon's Test Cases

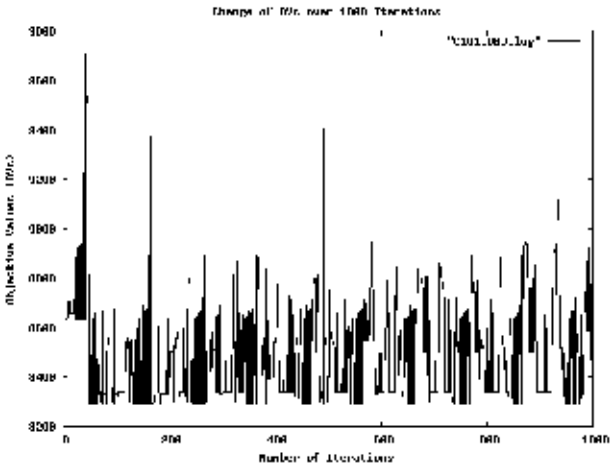
Besides, VVH+* searches outperform the corresponding PFIH+* searches in 2 out of 3 test cases on the average. This improvement can be explained in term of the additional domain knowledge gained from the parameter v_p which is the number of vehicles used in the best-known results. Nevertheless, VVH+* searches fail to find solutions in some specific test cases probably due to the bound $v_p + 1$ or $v_p + 2$ imposed on the initial search, which may hinder the applicability of this initialisation heuristic in many real-life applications. After all, our proposed GLS-TS is shown to be effective with both the PFIH and VVH. To have a

[2] It should be noted that Solomon's test cases [10] have been tackled by researchers for decades, thus leaving very little room for improvement. In addition, the original TS [1] and GLS [3] methods already achieved very good results. Therefore, an improvement of 0.13% over the best published result can be impressive and significant relative to the performance of all other solvers in the area.

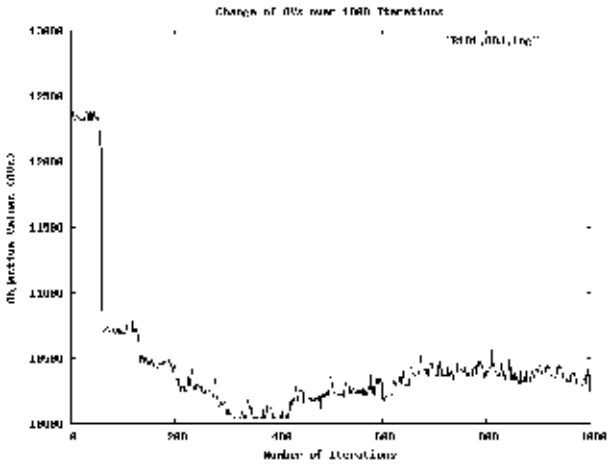
deeper understanding on their actual search behaviour, the variations of the returned objective values over 1,000 iterations are shown for both GLS-TS solvers integrated with the PFIH and VVH on the selected cases of C101, R101 and RC101 in Figure 9 (a) − (f) respectively. In general, the clustered distribution of customer requests as in the case of C101 presents much challenge to both GLS-TS solvers integrated with the PFIH and VVH with many spikes shown in their search trajectories in Figure 9(a) and 9(b). Apparently, the search is bounded by the best objective value (around *8,300*), that should be very close to the global optimum, found by either solver in this case. Therefore, the search fails to go beyond this very best objective value even after *1,000* iterations. Another possible extension is to implement an intelligent monitoring mechanism that should prompt the users for continuation or not whenever the search trajectory reveals that the search cannot exceed certain 'threshold' value for a long period of time.



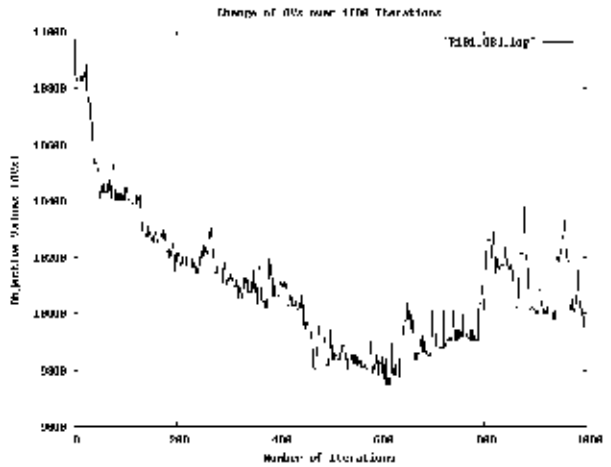
(a) Change of Obj. Values for PFIH+GLS-TS on C101



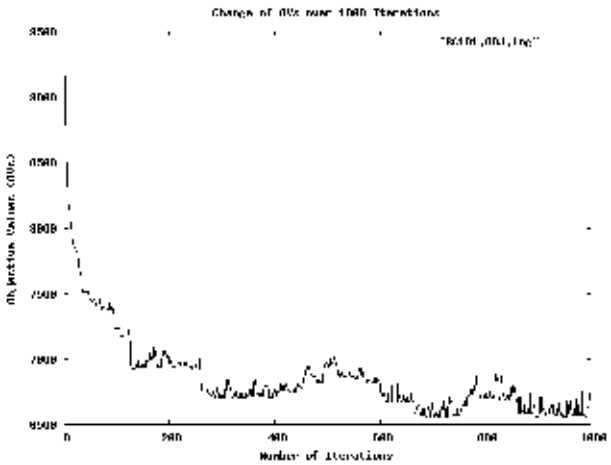
(b) Change of Obj. Values for VVH+GLS-TS o



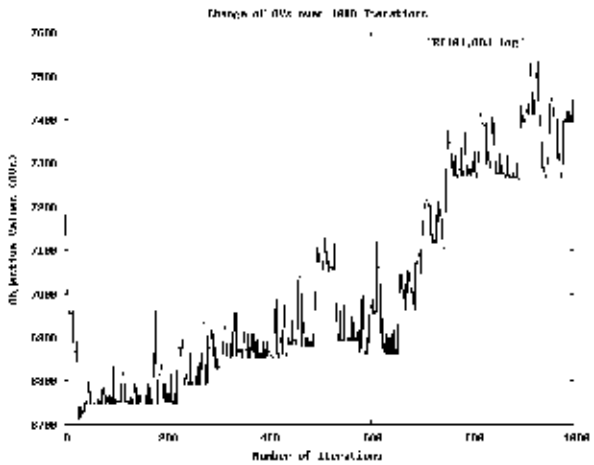
(c) Change of Obj. Values for PFIH+GLS-TS on R101



(d) Change of Obj. Values for VVH+GLS-TS on R101



(e) Change of Obj. Values for PFIH+GLS-TS on RC101



(f) Change of Obj. Values for VVH+GLS-TS on RC101

Figure 9. The Potential Gains for the GLS–Tabu Search (GLS-TS) integrated with the PFIH or VVH on the selected cases of C101, R101 and RC101

Figure 9(c) and 9(d) reveals the search behaviour of the PFIH+GLS-TS and VVH+GLS-TS solvers on the R101 problem with a random distribution of customer requests. Both solvers show an overall trend of gradually minimizing the best objective values returned, with their lowest objective values obtained at around *400* or *600* iterations respectively, while demonstrating some rebounds in their returned objective values in the remaining part of the search process. Lastly, Figure 9(e) and 9(f) gives the search behaviour of the PFIH+GLS-TS and VVH+GLS-TS solvers on the RC101 problem with both clustered and random distribution of customer requests. The PFIH-GLS-TS solver shows a similar pattern of search trajectory with the lowest objective value obtained at about *700* iterations. Interestingly, after attaining the lowest objective at the first hundreds of iterations, the VVH-GLS-TS solver shows an overall trend of gradually "increasing" the lowest objective values returned over the residual part of the search process. This is probably due to the penalty scheme used in the GLS search that continuously penalizes some particular features over the iterations, thus leading to the phenomenon of "over-learning" that causes the best objective value returned to deviate further from the minimal objective value attained. In general, the objective values of the initial solutions obtained by the VVH are always better than those returned by the PFIH, thus demonstrating the effectiveness of the VVH in solving this challenging set of vehicle routing problems. After all, the plotting is obtained from some selected results for which their specific patterns of search behaviour may not be exhibited in other problems of the same class since each individual problem may have its unique set of features, that prompts for our further investigation.

In addition, VVH-GTS scored the lowest sum of $TV \times TD$ among the compared solvers probably due to the probabilistic nature of the non-targeted states, that are the non-*X* states as discussed in Section 4.2, which *accidentally* lead the search to the solution states in some specific instances. The actual reason(s) for the overall improvement on the sum of $TV \times TD$ in this particular case require a more detailed investigation. Nevertheless, this result clearly demonstrates the effectiveness of combining both GLS and TS meta-heuristics in solving real-life VRP-TWs.

Furthermore, the sum of $TV \times TD$ for the VVH+GLS-TS-SA is the largest among the VVH+* solvers while the PFIH-GLS-TS-SA shows the largest averaged deviations from the best results among all the comparing solvers, clearly demonstrating the difficulty in controlling the local search or Tabu search with the integrated simulated annealing technique. We will try to explain this difficulty in term of the probabilistic model we have discussed in Section 4.2. Roughly, integrating the simulated annealing technique to decide which search method to use in each iteration is similar to adding a pre-requisite state before each original state in the state transitions of the TS algorithm as shown in Figure 6(a). Each pre-requisite state has two out-going arrows as two possible transitions – one leads to the original TS state while another leads to the additional local search state as possibly occurred in the GLS algorithm. The overall effect is the main component ($p_{ab} \cdot p_{bc.} p_{cd}$) of the probability of successfully reaching the solution state is significantly lowered by a factor of h^n where h, ranging from *0 to 1,* is the probability of still being the most rewarding search method (with respect to the normal local search method), and n is the number of TS states involved. Moreover, the newly added pre-requisite and local-search states into the original state transition diagram of the TS simply means the resulting GLS-TS-SA algorithm more difficult to manage with more possibility to consider. After all, this observation prompts us to carefully refine the simple performance measure used in our proposed GLS-TS-SA solver for testing.

Lastly, we try to identify the best optimiser for each of the 6 classes of Solomon's test cases. We observe for the C1 and C2 classes, where there is no clear winner in term of the sum of $TV \times TD$, PFIH+GLS is the simplest and most stable solver. Also, it is worth noting that GLS-TS-SA is marginally better in the C1 class with only *0.14%* deviated from the best published results. In addition, VVH+GLS-TS performs the best for both R2 and RC2-type problems. For a more detailed comparison on the performance of the different solvers in each individual problem class of the Solomon's test set, refer to [17].

6. Concluding remarks

In this chapter, we proposed our 2 interesting search hybrids by integrating the well-known guided local search (GLS) and Tabu search (TS) algorithms to effectively solve the VRPs. As opposed to the GTS in [1], we consider a different integration of Tabu Search into GLS framework as GLS-TS in which the GLS-type penalty scheme is invoked only at stable states. More interestingly, from GLS-TS, we implement the SA approach to work as the guiding principle to intelligently choose which is the *right* search method to apply in each iteration based on some cooling schedules. The cooling schedule is governed by a standard logistic function as defined in [11] together with some performance metric such as the reduction in the total distance travelled. There was some previous work which combined simulated annealing with Tabu search [11]. However, in our work, the SA mechanism is embedded into the combined GLS and Tabu search framework to decide which search method to apply in each search step.

In the empirical evaluation of our proposals on the Solomon's benchmarks, we obtained exciting results from our proposed GLS-TS algorithm. Generally speaking, GLS-TS is a very efficient algorithm compared to its ancestors, the GLS and GTS. Surprisingly, the VV+GLS-TS produces a better-than-best-published result for the R205 problem of the Solomon's test set. In addition, from our empirical results, we made some careful observations that improving on TV through any heuristic search operator can be more rewarding than the

improvement on *TD* through the GLS-type penalty scheme on the long arcs since the objective function is $TV \times TD$. This will definitely provide one exciting direction of our on-going research work. More importantly, we provide a simple and interesting probabilistic model for analysing the search behaviour of our proposed hybrids. This readily forms the basis for our future investigation.

Our new approach in the hybridisation of GLS and TS algorithms with simulated annealing truly opens up many new directions for future exploration. Here, we suggest to use the simulated annealing as a more sophisticated guiding principle, or namely meta-meta-heuristic, to monitor the underlying meta-heuristics such as GLS which in turn controls the search on the initial solution returned by the initialisation heuristics such as the PFIH or VVH. For the future extension, we should apply our proposed hybrid approaches to new problem sets so as to demonstrate the strength of the different search strategies. Besides, we will look into other types of initialisation method to be integrated into our search hybrids to improve their performance. Another interesting direction is to investigate other possible logistic functions and performance metrics to particularly improve the performance of our proposed GLS-TS-SA solver. Lastly, it should be interesting to investigate other TS-related variants integrated with the simulated annealing such as an adaptive strategy which can flexibly invoke the GLS penalty scheme within the TS framework depending on its relative merit.

7. Acknowledgements

The authors wish to thank Professor Edward Tsang and Professor Yi Shang for their fruitful discussions. In addition, the authors are grateful to the generous supports from the Department of Electrical and Electronic Engineering, the University of Hong Kong to this work.

8. References

Bruno De Backer, Vincent Furnon, Philip Kilby, Patrick Prosser and Paul Shaw. Solving Vehicle Routing Problems using Constraint Programming and Meta heuristics. Journal of Heuristics. Vol 1-16(1997), Kluwer Academic Publisher, 1997.

Philip Kilby, Patrick Prosser and Paul Shaw. Guided Local Search for the Vehicle Routing Problems. *Proceedings of 2nd International Conference on Metaheuristics* (MIC97). Pp.1-10, Sophia-Antiplois, France, July 21-24,1997.

E. Tsang, Chang J. Wang, Andrew Davenport, C. Voudouris and Tung L.L.. *A family of Stochastic Methods for Constraint Satisfaction and Optimisation*. The First International Conference on the Practical Application of Constraint Technologies and Logic Programming, London, April 1999

Ben. Wah and Y.X. Chen. *Constrained Genetic Algorithms and their Applications in Nonlinear Constrained Optimization.* Proceedings of IEEE ICTAI. Vancouver, Cananda, November 2000.

I.H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Ann. Oper. Res.*. 41:421-451,1993.

J.C. Jee, *Solving Vehicle Routing Problems with Time Windows Using Micro-Genetic Algorithms,* Proceedings of the 6th National Ungergraduate Research Opportunities Programme Congress 2000, Faculty of Engineering, NUS, September 7-8, 2000.

P. Prosser and P. Shaw. *Study of greedy search with multiple improvement heuristics for vehicle routing problems.* Technical Report RR/96/201, Department of Computer Science, Univeristy of Strathclyde, Glasgow, January 1997.

Rochat, Y. and Taillard, E. D.. *Probabilistic diversification and intensification in local search for vehicle routing*. Technical report CRT-95-13, Centre de recherche sur les transports, Université de Montréal, Montréal. 1995.

D., Seah . *Ant Colonoy Optimization On Vehicle Routing Problems.* Thesis for the Master of Computer Science, National University of Singapore, 1999

M.M. Solomon. *Algorithms for the vehicle routing and scheduling problem with time windows. Oper. Res.* 35:254-265,1987.

William M. Spears. *Simulated Annealing for Hard Satisfiability Problems*. NCARAI Tech. Report AIC 93-015. AI Center, Naval Research Laboratory, Washington D.C., 1993.

Chris Voudouris and Edward Tsang. *Technical Report CSM-247*, Department of Computer Science, Univeristy of Essex, August 1995

M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. Management Science, 40(10):1276-1290, 1994.

S. Lin. Computer solutions for the traveling salesman problem. Bell Systems Technology Journal, 44:2245-2269, 1965.

M. W.P. Savelsbergh. Computer aided routing. Centrum voor Wiskunde en Informatica, Amsterdam, 1988.

Paul G. Hoel. Elementary Statistics (4th Edition). John Wiley & Sons Inc., 1976.

K.T. Ma. Using Hybrid Search Methods to Solve A.I. Search and Optimization Problems. Hons Thesis, The National University of Singapore, March 2001.

4

A Hybrid Ant Colony System Approach for the Capacitated Vehicle Routing Problem and the Capacitated Vehicle Routing Problem with Time Windows

Amir Hajjam El Hassani, Lyamine Bouhafs and Abder Koukam
Systems and Transportation Laboratory
University of Technology of Belfort-Montbliard,
France

1. Introduction

The Vehicle Routing Problem (VRP) is a class of well-known NP-hard combinatorial optimisation problems. The VRP is concerned with the design of the optimal routes, used by a fleet of identical vehicles stationed at a central depot to serve a set of customers with known demands. In the basic version of the problem, known as a Capacitated VRP (CVRP), only capacity restrictions for vehicles are considered and the objective is to minimize the total cost (or length) of routes.
The Capacitated Vehicle Routing Problem with Time Windows (CVRPTW), which is a generalization of the CVRP, is one of the most studied variants of the VRP. In the CVRPTW, the vehicles must comply with constraint of time windows associated with each customer in addition to capacity constraints.
The study of the VRP is very important. The VRP contributes directly to a real opportunity to reduce costs in the important area of logistics. Logistics can be roughly described as the delivery of goods from one place (supplier) to others (consumers). Transportation management, and more specifically vehicle routing, has a considerable economical impact on all logistic systems.
Due to the nature of the problem, it is not viable to use exact methods for large instances of the VRP. Therefore, most approaches rely on heuristics that provide approximate solutions. Some specific methods have been developed to this problem. Another option is to apply standard optimization techniques, such as tabu search, simulated annealing, constraint programming, genetic algorithms and ant systems.Our main interest is about the metaheuristics used to solve the VRP and more particularly about the ant colony system.
The first algorithm based on the ant colony system, applied to the CVRP, was proposed by [Bullnheimer & al. 1999] known as « Ant System » (AS), applied first for the TSP in [Dorigo & al. 1996]. The pheromone and the nearest neighbor heuristic are used to build the routes of the vehicles. To improve the routes, a heuristic of 2-OPT is combined with the AS. This algorithm was tested on 40 benchmark but the performances are inferior to those relative to the tabu search algorithm. Yet, in terms of time accomplishment, the AS is a serious rival for the existing metaheuristics.

The same authors [Bullnheimer & al. 1999] proposed an improved AS which consists of replacing the nearest neighbor heuristic, in the transition rule of the basic algorithm (AS), by the Savings heuristics of [Paessens 1988]. The same process used in their first algorithm is applied for the pheromone updating rule. At the end of each iteration, a local search heuristic (2-OPT) is applied to improve the paths/routes. A second local search heuristic is used in the choice of clients. For each client, the distances are sorted out in an increasing manner ; the list of clients not yet visited. The tests confirm an improvement in the results in comparison to the first version of the algorithm of [Bullnheimer & al. 1999]. However, the algorithm is still limited ; the best solutions published are rarely achieved.
The algorithm proposed by [Doerner & al. 2002] is almost identical to the algorithm proposed by [Bullnheimer & al. 1999]. The only différence is that the algorithm of [Doerner & al. 2002] combines the AS with the Savings heuristic of [Clarke & Wright 1964]. The results obtained from this algorithm do not improve in a significant manner those of the previous approaches.
Contrasting to all the methods based upon "Ant System", we have proposed an approach, [Bouhafs & al. 2004], based on the new version of the « Ant Colony System» [Dorigo & Gambardella 1997] proposed to improve the «Ant System» and namely for the problems with large number of instances.
In this chapter, we propose an improvement of our approach [Bouhafs & al. 2004] by integrating an algorithm of Simulated annealing to the pheromone updating rule of the Ant Colony System.
In Section 2 we present the formulation of the CVRP. Section 3 deals with the description of the algorithm proposed to solve the CVRP. Section 4 describes the experimental results for the CVRP. Section 5 presents the formulation and description of the algorithm proposed to solve the CVRPTW. Section 6 describes the experimental results for the CVRPTW and Section 7 concludes the works.

2. CVPR formulation

The CVRP can be represented as a weighted directed graph $G = (V, A)$ where $V = \{v_0, v_1, v_2, ..., v_n\}$ represents the set of the vertices and $A = \{(v_i, v_j) : i \neq j\}$ represents the set of arcs.

The vertex v_0 represents the depot and the others represent the clients. To each arc (v_i, v_j) a non negative value d_{ij} is associated. This value corresponds to the distance between the vertex v_i and the vertex v_j in terms of cost or time between the two vertices. A demand q_i and time service δ_i $(q_0 = 0, \delta_0 = 0)$ are associated with each client (vertex) v_i .
In this case, the objective is to minimize the total cost of routing and at the same time respect the following constraints: (1) Every client is visited exactly once by exactly one vehicle , (2) all the vehicles paths/routes start and end at the depot, (3) the total demand of clients of each path/route should not exceed the capacity of each vehicle.
The number of vehicles is supposed to be illimited ; it is calculated during the construction of the routes of vehicles.
The figure 2.1 shows the graphic representation of a CVRP example.

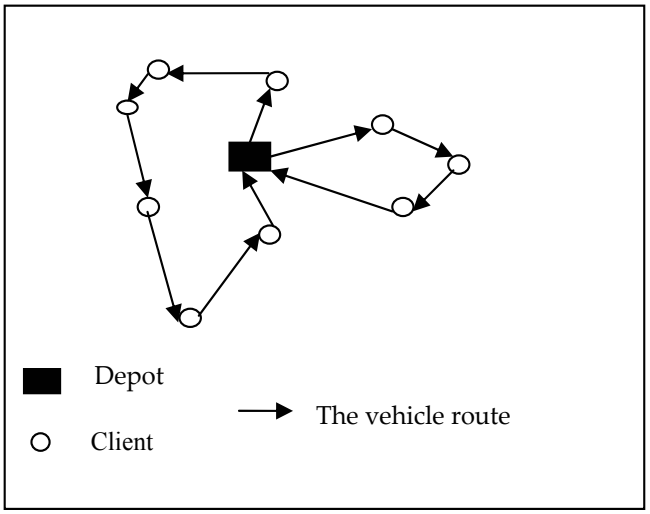


Figure 2.1 Example of CVRP solution

Here, we give the CVRP formulation found in literature. But, before that, we describe the list of variables that will help us in the formulation :

- $G = (V, A)$
- $V = \{v_0, v_1, v_2, ..., v_n\}$ where v_0 represents the depot and $v_1, ..., v_n$ all the clients
- q_i the demand of the client i, $i \in V$
- d_{ij} the distance (cost) between the vertices(clients) v_i and v_j
- $K = \{k_1, k_2, ..., k_m\}$ represents the vehicles fleet
- Q the capacity of each vehicle $k_i \in K$ (the fleet is homogeneous)

In order to find the clients visit order, we define the decision variables as follows :

$$x_{i,j}^k = \begin{cases} 1 & \text{if the vehicle k visits the client j directly after the client } i \\ 0 & \text{otherwise} \end{cases}$$

$$y_i^k = \begin{cases} 1 & \text{if the client } i \text{ is served by the vehicle } k \\ 0 & \text{otherwise} \end{cases}$$

The objective function is

$$Min \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} d_{ij} x_{i,j}^k \tag{2.1}$$

with the constraints

$$\sum_{k \in V} \sum_{j \in N} x_{i,j}^k = 1, \quad \forall i \in V \tag{2.2}$$

$$\sum_{j \in V} x_{i,j}^k - \sum_{j \in V} x_{j,i}^k = 0, \ \forall i \in V, k \in K \tag{2.3}$$

$$\sum_{j\in V} x_{0,j}^{k} = 1\ ,\ \forall k \in K \tag{2.4}$$

$$\sum_{j\in V} x_{j,n+1}^{k} = 1\ ,\ \forall k \in K \tag{2.5}$$

$$x_{i,j}^{k} = 1 \Rightarrow y_i - q_j = y_j\ ,\ \forall i, j \in V, k \in K \tag{2.6}$$

$$y_0 = Q,\ \ 0 \le y_i\ ,\ \forall i \in V \tag{2.7}$$

$$x_{i,j}^{k} \in \{0,1\}\ ,\ \forall i, j \in V, k \in K \tag{2.8}$$

The function of the euclidean cost solution $X = \left(x^{k}_{ij}\right)\ \ \forall i, j \in V, k \in K$ is defined by :

$$Cost(X) = \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} d_{ij} x_{i,j}^{k} \tag{2.9}$$

The number of vehicles used by the solution X is defined by

$$\text{NB vehicles}\ (X) = \sum_{k\in K}\sum_{i\in V} x_{0,j}^{k} \tag{2.10}$$

3. A hybrid algorithm based on ant colony system and annealing for the capacited vehicle routing problem (CVRP)

Our algorithm is based on a hybrid ACS with an algorithm of Savings and a local search (2-opt). In the transition rule, we introduce the Savings heuristic that allows an enrichment of the search and to calculate the utility to combine 2 clients in the same route or to put them in two different paths/routes. The 2-opt is applied at each definition of the routes by the ants, juste before the global updating rule of pheromones. The simulated annealing intervenes in the phase relative to the updating of the pheromones where two rules are applied as in « the Ant Colony System » (ACO) introduced in [Dorigo & Gambardella 1997]. Before developing the different steps of the algorithm, we explain first the 2 heuristics 2-opt and Savings used in our approach.

3.1 The 2-OPT heuristic

The principle of the 2-OPT is to delete 2 arcs of the same route and to replace them by 2 other arcs in order to improve the cost of this route and to delete the road junctions.
In the figure 3.1, the two scattered arcs are deleted in the left graph. They are replaced by the two arcs labeled (A and B) in the right graph in order to find a new route. The direction of some arcs in directed graphs, that are not concerned by a 2-OPT operation, can be modified during the construction of the new route.

3.2 The savings heuristic

This heuristic was proposed by [Clarke & Wright 1964] and improved by [Paessens 1988]. It is the basis of most of the commercial software used to solve the vehicle routing problems in the industrial applications. The objective of this heuristic is to determine whether it is better

to combine the clients v_i and v_j in the same route (when the value of γ_{ij} is big) or to put them in two different routes. The Savings value of the clients v_i and v_j is calculated as follows :

$$\gamma_{ij} = d_{i0} + d_{0j} - g.d_{ij} + f.\left|d_{i0} - d_{0j}\right|$$

where

- d_{ij} represents the distance between the vertice i and the summit j,
- The index 0 corresponds to the depot
- g and f represent 2 parameters of the heuristic.

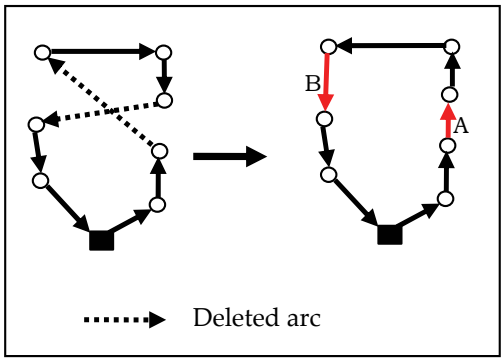


Figure 3.1 The heuristic principle of local search 2-OPT

3.3 Description of the algorithm

3.3.1 Construction of the routes/paths

The algorithm ACS of [Dorigo & Gambardella 1997], presented for the first time to solve the TSP, takes into consideration the pheromone and the heuristic of the nearest neighbor for the construction of the routes. Our approach ACS hybrid, presented here, takes into consideration the Savings heuristic as well for such construction.

Initially in the ACS hybrid, m ants are positioned on all the graph vertices and a quantity of initial pheromone is applied on the arcs. Each ant takes its departure from the depot to visit the clients. Each client is visited once and only once by an ant, however, the depot can be visited several times. If the load stored by the ant exceeds the vehicle constraint capacity, the ant must return to the depot. We then get a complete route for a vehicle. When an ant goes back to the depot, it starts from scratch again. It initializes another route to visit other new clients. This operation is repeated over and over again until all clients are visited. This means that a solution to the Capacitated Vehicle Routing Problem (CVRP) has been found.

During the process of building a route, the ant modifies the quantity of pheromone on the chosen arc by applying a local updating rule. Once all the ants are done with the building of their routing, the quantity of pheromone on the arcs belonging to the best routing found is updated according to the global updating rule.

The rule used for the construction of the routes is described hereafter. An ant k, positionned on a vertice i, chooses the next vertice j to visit by applying the probabilistic rule $p_k(i, j)$ given in the equations (3.1) and (3.2).

$$j = \begin{cases} \arg\max_{u \in F_k(i)} \left\{ (\tau_{iu})^{\alpha} . (\eta_{iu})^{\beta} . (\gamma_{iu})^{\lambda} \right\} & if\ q \le q_0 \\ J & if\ q \succ q_0 \end{cases} \quad (3.1)$$

J is a random variable generated according to the function of distribution given by :

$$p_{ij} = \begin{cases} \dfrac{(\tau_{ij})^{\alpha} . (\eta_{ij})^{\beta} . (\gamma_{ij})^{\lambda}}{\sum_{u \in F_k(i)} (\tau_{iu})^{\alpha} . (\eta_{iu})^{\beta} . (\gamma_{iu})^{\lambda}} & \text{if } u \in F_k(i) \\ 0 & \text{otherwise} \end{cases} \quad (3.2)$$

where

- $F_k(i)$ is the list of the vert not yet visited by the ant k positioned at the vertex i,
- q is a random variable that follows a uniform distribution on $[0,\ 1]$,
- q_0 is a parameter $(0 \le q_0 \le 1)$ that determines the relative importance of the exploitation versus the exploration. Before an ant visits the next vertex, q is generated randomly. If $q \le q_0$, the exploitation is then encouraged, otherwise the process of exploration is encouraged,
- τ_{ij} is the quantity of pheromone associated to the arc (i, j)
- η_{ij} is the heuristic of visibility that is the inverse of the distance between the vertices i and j,
- $\gamma_{ij} = d_{i0} + d_{0j} - g.d_{ij} + f.\left|d_{i0} - d_{0j}\right|$ is the heuristic of Savings where f and g are two parameters.

α, β and λ are three parameters that determine the relative importance of the pheromone, the distance and Savings respectively.

3.3.2 Pheromones updating

The originality in our algorithm is to introduce the simulated annealing in the phase of the pheromones updating. While an ant is building its solution, the level of pheromone on each arc (i, j) visited is modified according to the following updating local rule :

$$\tau_{ij}^{new} = (1 - \rho)\tau_{ij}^{old} + \rho \Delta \tau_{ij} \quad (3.3)$$

where

ρ is the parameter of evaporation $(0 < \rho < 1)$

$\Delta \tau_{ij} = \tau_0$ is the quantity of initial pheromone.

Once each ant has defined its tours, a 2-opt local search is applied in order to improve the vehicles routes. This heuristic is applied at the end of each iteration and just before the global updating rule of the pheromones.

In the algorithm ACS, the ant allowed to lay down a quantity of pheromones to guide the search is the one that has found the best solution.

In our algorithm ACS, each ant that finds a solution S' where

$\Delta S(= Cost(S') - Cost(S)) \prec 0$ or $\exp(-\Delta S / T) \succ \mu$ is given permission to lay down a quantity of pheromone.

where

- S is the best current solution
- μ is a random variable that follows a uniform distribution on $[0,1]$
- $Cost(S)$ is the cost of the solution S,
- $\exp(-\Delta S / T)$ is the criteria of Metropolis used for the simulated annealing
- T parameter that represents the temperature in the simulated annealing

The integration of this choice criterion in the global updating rule permits to accept some solutions which are not inevitably better than the current solution, and hence to diversify the search.

The overall updating rule is given in the equation (3.4)

$$\tau_{ij}^{new} = (1-\rho)\tau_{ij}^{old} + \rho\Delta\tau_{ij} \quad (3.4)$$

Where ρ is the parameter of evaporation of the pheromone $(0 < \rho < 1)$

If the arc (i, j) is used by an ant whose solution is accepted (meets the criteria defined above), the quantity of pheromone is then increased on this arc by $\Delta\tau_{ij}$ that is equal to $1/L^*$ with L^* is the length of tours found by this ant.

4. Experimentations and results

After the description of the ACS hybrid algorithm, we evaluate its performance by considering a set of instances relative to the CVRP problem. The algorithm is coded in java language and executed on a laptop PC, equipped with Windows XP system, a P4 processor at a 2.4 GHz speed, a RAM of 512 Mo. Our experimental study was about a set of instances Of different sizes. These instances are detailed in the Table 4.1. They can be grouped in 3 categories according to their originators. instances (A, B et P) : [Augerat & al. 1995], instances E : [Christofides & Eilon 1969], instances C : [Christofides & al. 1979].

For each instance, we give the number n of clients and the capacity Q of a vehicle.

4.1 The algorithm parameters adjustment

We initialize the algorithm parameters by the following values :

- The number of ants $m = 10$ initially placed at random on all the summits,
- $\alpha = 1,\ \beta = 2$ with α, β and λ are three parameters that determine respectively the relative importance of the pheromone, of the distance and of Savings,
- $\rho = 0.1$ is the parameter of evaporation of the pheromone used in updating rules of the pheromones,
- $f = g = 2$ with f and g the two parameters of the heuristic of Savings,

- $q_0 = 0.75$ is the parameter that determines the relative importance of the exploitation versus the exploration. Before that an ant visits the next summit, q is generated randomly. If $q \le q_0$ the exploitation is then encouraed, otherwise it is the exploration process that is encouraged,
- $\tau_0 = 10^{-6}$ represents the initial quantity of the pheromone put down on all the arcs.
- $T = 100$ represents the initial value of the temperature,
- Initially the parameter of the temperature T is T_0 $(T \leftarrow T_0)$,
- $\theta = 0.9$, at each iteration, the temperature is modified by the formula $(T \leftarrow \theta * T)$.

Instance	n	Q	Instance	n	Q
B-n68-k9	68	100	C1	50	160
B-n78-k10	78	100	C2	75	140
B-n66-k9	66	100	C3	100	200
B-n50-k8	50	100	C4	150	200
B-n57-k9	57	100	C5	199	200
B-n63-k10	63	100	C11	120	200
B-n67-k10	67	100	C12	100	200
P-n51-k10	51	80	A-n63-k9	63	100
P-n76-k5	76	280	A-n63-k10	63	100
E-n76-k10	76	140	A-n69-k9	69	100
E-n76-k15	76	100	A-n80-k10	80	100
A-n60-k9	60	100			

Table 4.1 details relative to the intances of tests (n : number of clients, Q : capacity of a vehicle)

4.2 Tests results

In this paragraph, we present a study of the tests performed of the ACS hybrid algorithm we have proposed. The tests results for the instances detailed in the table 4.1 are given in the table 4.2. The first column of the table represents the name of the instances dealt with. The second column specifies the best solution known found by the heuristics (Dist : distances & Nbv : Number of vehicles). The column with the header H_ACS gives the solution found by our approach. Finally, the last column gives the deviation of our solution in comparison with the best solution published found by other heuristics. It is calculated by the following formula :

$$deviation = \frac{Solution(H\_ACS) - Solution(best\ known)}{Solution(best\ known)} * 100$$

From table 4.2, we note that the results found by our approach have a deviation average of (0.022%) in comparison with the best solutions known regarding all the tests. What explains this point is that the solutions found by our ACS hybrid are very close to the best solutions published so far. Moreover, our method allows even an improvement of certain solutions relative to some instances (B-n68-k9, B-n78-k10, B-n66-k9, A-n60-k9, P-n51-k10).

We also draw a comparison between our approach and that of [Bullnheimer & al. 1999] to evaluate the quality of our solution and the CPU time. The tests of comparison are done on the benchmarks of [Christofides & al. 1979], and presented in the table 3.1 (C1, C2, C3, C4, C5, C11, C12). At the initialization of this approach, N ants are placed on the graph summits, N represents thus the number of clients. For each instance of the problem, the algorithm is executed $2 * N$ iterations.
The results of comparison are given in table 4.3. The first column of the table represents the name of the instances dealt with. The column "best publ." represents the best solution known in literature. The third column specifies the best solution found by AS of [Bullnheimer & al. 1999] (COST: cost of the solution, CPU: computing time in seconds and Dev : deviation in comparison with the best solution known). The column with the header H_ACS gives the solution found by our approach after 500 iterations for each instance.

Instance	Best known		H_ACS		deviation
	dist.	nb v.	dist.	nb v.	
B-n68-k9	1304	9	**1300.91**	9	-0,23%
B-n78-k10	1266	10	**1238.28**	10	-2,19%
B-n66-k9	1374	9	**1371.67**	9	-0,17%
B-n50-k8	1313	8	1317.34	8	0,30%
B-n57-k9	1598	9	**1598.0**	9	0%
B-n63-k10	1537	10	1540.71	10	0,19%
B-n67-k10	1033	10	1035.46	10	0,19%
A-n60-k9	1408	9	**1357.70**	9	-3,57%
A-n63-k9	1634	9	1636.94	9	0,18%
A-n63-k10	1315	10	1322.93	10	0,60%
A-n69-k9	1168	9	1177.36	9	0,80%
A-n80-k10	1764	10	1787.05	10	1,30%
P-n51-k10	745	10	**743.26**	10	-0,23%
P-n76-k5	631	5	631.0	5	0%
E-n76-k10	832	10	836.37	10	0,52%
E-n76-k15	1032	15	**1030**	15	-0,19 %

Table 4.2 CVRP experimentations results

Instance	**best publ.**	AS **Bullnheimer et al. 1999]**			**H_ACS**		
		cost	cpu(s)	dev.	cost	cpu(s)	dev.
C1	524.61	524.61	6	0%	524.61	2	0%
C2	835.26	844.31	78	1.08%	838.86	20	0.43%
C3	826.14	832.32	228	0.75%	834.77	35	1.04%
C4	1028.42	1061.55	4048	3.22%	1035.23	80	0.66%
C5	1291.45	1343.46	5256	4.03%	1304.25	200	0.99%
C11	1042.11	1065.21	552	2.22%	1046.81	32	0.45%
C12	819.56	819.56	300	0%	819.56	30	0%

Table 4.3 Test of comparison between the ACS hybrid algorithm and the AS algorithm of [Bullnheimer & al. 1999]

Through Table 4.3, we deduct that the average deviation of our approach (0,51%), in comparison with the best solutions known, is smaller than the approach of [Bullnheimer & al. 1999] (1.61%). This proves that our approach gives better performanes. Moreover, the solutions are found in far less computing time than [Bullnheimer & al. 1999]. However, we are unable to make a direct comparison in terms of computing time because the approaches are tested on different machines.
To sum up, our approach finds solutions of good qualities within reasonable computing time.

5. The CVRPTw (The Capacitated Vehicle Routing Problem With Time Windows)

The CVRPTV is one of the most studied variants of the vehicle routing problems. We can say that the CVRPTW is a generalization of the CVRP. In fact, in addition to the capacity constraints introduced in the CVRP, the vehicles must respect the constraints of time windows associated with each client.

5.1 The CVRPTV formulation

As the CVRP, the CVRPTW can be represented by a weighted directed graph $G = (V, A)$, where

- $V = \{v_0, v_1, v_2, ..., v_n\}$ represents the whole set of vertices,
- $A = \{(v_i, v_j) : i \neq j\}$ represents all the arcs between the vertices.

The vertex v_0 represents the depot and the others represent the clients. For each arc (v_i, v_j) is associated a non negative value d_{ij} that corresponds to the distance between the vertex v_i and the vertex v_j that is the cost between the two vertices.

Each client (vertex) v_i has

- A demand q_i $(q_0 = 0)$
- A time window, which means that each client *i* has a predefined time with an earliest arrival time e_i (lower bound) and a latest arrival time l_i (upper bound) for the visit of vehicles.

The vehicles must arrive before the upper bound of the window l_i and if they arrive before the lower bound e_i they must wait awhile w_i (time wait). Each client imposes a service time δ_i that corresponds to the goods loading/unloading time. The time window of the depot means that each vehicle that leaves the depot at a time e_0 go back to the depot before the time l_0. We suppose that the vehicles fleet is homogenous and that the capacity of each vehicle is Q.

The objective function of the CVRPTW is to minimize the total cost of routing (the distance or the total time traveled) while respecting the constraints concerning the CVRP as well as those of time windows. The graphical representation is the same as the CVRP.

5.2 ACS improved for the CVRPTW

Our algorithm for the CVRPTW is an algorithm of hybrid optimization. It is based on ACS and local searches. In addition of the heuristic of Savings introduced for the CVRP, we use a new local search in order to take into account informations specific to the CVRPTW. We name this heuristic a wait heuristic that favors the clients choice whose wait time is smaller. It is calculated by the following formula :

$$\omega_{ij} = 1/w_j \tag{5.1}$$

and

$$w_j = \begin{cases} e_j - t_j & if \quad (e_j - t_j) > 0 \\ 1 & \text{otherwise} \end{cases} \tag{5.2}$$

where

- e_j is the time of arrival as earliest to the client j (Time Windows lower bound of client j)
- t_j is the actual arrival time to client j

Once the ants find routes, we improve them by a local search heuristic $2-opt^*$ whose objective, contrasting to the 2-opt heuristic that tries to improve one route, is to improve 2 routes in the same routing. A $2-opt^*$ operation consists of deleting 2 arcs from 2 different routes (one bone from each route/path) to divide each route in two parts and exchange the second parts of the 2 routes/paths while respecting the constraints of both the time windows and the capacities (Figure 5.1).

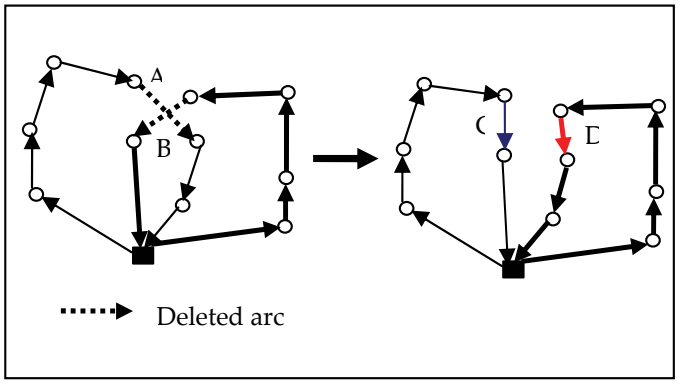


Figure 5.The local search heuristic 2-OPT *

Each ant's point of departure is the depot to visit the clients. Each client is visited once and only once by an ant, however, the depot can be visited several times. If the load stored by the ant exceeds the vehicle constraint capacity or violates the constraint of time windows of the depot, the ant must return to the depot. We then get a complete route for a vehicle. When an ant goes back to the depot, it starts from scratch again in terms of its accumulated load and time. It initializes another route to visit other new clients. This operation is repeated over and over again until all clients are visited. This means that a thorough

solution to the Capacitated Vehicle Routing Problem Time Windows (CVRPTW) has been found.

When an ant k is localized to client i, the transition rule for the client j is given as follows.

$$j = \begin{cases} \arg\max_{u \in F_k(i)} \left\{ (\tau_{iu})^{\alpha} . (\eta_{iu})^{\beta} . (\gamma_{iu})^{\lambda} . (\omega_{iu})^{\theta} \right\} & if\ q \le q_0 \\ J & if\ q \succ q_0 \end{cases} \tag{5.3}$$

J is a random variable generated according to the following function of distribution

$$p_{ij} = \begin{cases} \dfrac{(\tau_{iu})^{\alpha} (\eta_{ij})^{\beta} (\gamma_{ij})^{\lambda} (\omega_{ij})^{\theta}}{\sum_{u \in F_k(i)} (\tau_{iu})^{\alpha} (\eta_{iu})^{\beta} (\gamma_{iu})^{\lambda} (\omega_{iu})^{\theta}} & \text{if } u \in F_k(i) \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

where

- P_{ij} is the probability to select the client j,
- $F_k(i)$ is all the clients not yet visited by the ant k positioned to client i,
- q is a random variable that follows a uniform law on $[0,1]$ & q_0 is a parameter $(0 \le q_0 \le 1)$,
- $(\alpha, \beta, \lambda, \theta)$ represent the parameters that have an impact respectively on the relative importance of the pheromone level, of the distance, of Savings and the wait heuristic

When an ant k goes to client j, we use the following formula to update locally the track of pheromone on the arc (i, j):

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + (\rho)\tau_0 \tag{5.5}$$

As soon as an ant generates all the routes/paths (CVRPTW solution), the local search $2-opt^*$ is applied to improve the routes when possible.

Finally, an overall updating of the pheromone tracks – that consists in reinforcing the arcs related to the best solution found by one of the ants – is carried out by using the following formula:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho \Delta\tau_{ij}(t) \tag{5.6}$$

where

- ρ $(0 < \rho < 1)$ est le paramètre d'évaporation de la phéromone.
- ρ $(0 < \rho < 1)$ is the parameter of evaporation of the pheromone.

If the arc (i, j) is used by the ant that has found the best solution, the quantity of pheromone is increased on this arc by $\Delta\tau_{ij}$ that is equal to $1/L^*$ with L^* is the length of the routes found by the best ant.

6. Experimentation and results

Our algorithm has been tested on a classical set benchmark Solomon problems. Solomon has generated sets of VRPTW benchmark composed of six different problem types (C1, C2, R1, R2, RC1, and RC2) available from:
http://w.cba.neu.edu/~msolomon/home.htm, each data set contains between eight to twelve 100-node problems. The names of the six problem types have the following meaning. Sets C have clustered customers whose time windows were generated based on a known solution. Problem sets R have customers location generated uniformly randomly over a square. Sets RC have a combination of randomly placed and clustered customers. Sets of type 1 have narrow time windows and small vehicle capacity. Sets of type 2 have large time windows and large vehicle capacity. Therefore, the solutions of type 2 problems have very few routes and significantly more customers per route.
For the setting parameters we use 20 artificial ants, $\alpha = 1,\ \beta = 2,\ \lambda = 1,\ \theta = 0.75$, $\rho = 0.1$, $q_0 = 0.75$, and $f = g = 2$ (parameters of the Savings function). Table 6.1 gives the computational results for the test problems obtained by our approach.
The average deviation of our approach is 0,30% in comparison with the best solutions known. This deviation is very feeble because our algorithm finds in many cases the best value published. In addition, the solutions are found in a reasonable computing time. We believe that the modified ACS algorithm needs further refinements. In particular, the suitable values of algorithm parameters.

7. Conclusion

In this chapter, we have presented two new algorithms for the CVRP and the CVRPTW. The two algorithms are based on the hybrid ant colony with local searches. The experimental results we have displayed show the quality of solutions achieved by our approaches. The objective of this study is to show the efficiency of the ant colony algorithms for the vehicle routing problems.
This study has shown that the combining of the ant colonies with specific informations to the problem under study enables the finding of competitive results. The experimental results we have displayed show the quality of solutions achieved by our approach. The objective of this study is to show the efficiency of the hybrid ant colony algorithms for the vehicle routing problems. This study has shown us that by combining the ant colonies with the simulated annealing, competitive results are found.

8.References

Augerat P, Belenguer J M, Benavent E, Corberan A, Naddef D, Rinaldi G. Computational results with a branch and cut code for the capacitated vehicle routing problem. Technical report RR 949-M, University Joseph Fourier, Grenoble (1995).

Bouhafs, L., Hajjam, A., Koukam, A.: "A Hybrid Ant Colony System Approach for the Capacitated Vehicle Routing Problem", 4th International Workshop, ANTS 2004, volume 3172 de LNCS, pages 414–415. Springer, September (2004).

Bullnheimer, B., Hartl, R. F. and Strauss, Ch.: Applying the ant system to the vehicle routing problem. In: Voss, S., Martello, S., Osman, I. H. and Roucairol, C. (Eds.): Meta-

Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer, Boston (1999).

Bullnheimer, B., Hartl, R. F. and Strauss, Ch.: An improved ant system algorithmfor the vehicle routing problem. Annals of Operations Search 89 319–328, (1999).

Christofides, N., Eilon, S.: An algorithm for the vehicle dispatching problem. Operational Search Quarterly 20 (3), 309–318, (1969).

Christofides, N., Mingozzi, A., Toth, P.: The vehicle routing problem. In: Christofides, N., Mingozzi, A., Toth, P., Sandi, C. (Eds.), Combinatorial Optimization, Wiley, Chichester, pp. 315-338, (1979).

Clark, G., and Wright, J. W.: Scheduling of vehicles from a central depot to a number of delivery points. Operations Search 14, 568-581, (1964).

Dorigo, M., Maniezzo, V., and Colorni, A.: "Ant system: Optimization by a Colony of Cooperating Agents". IEEE Trans. Sys., Man, Cybernetics 26 1, pages 29-41, (1996).

Doerner, K., Gronalt, M., Hartl, R.F., Reimann, M., Strauss, C., and Stummer, M.: "SavingsAnts for the Vehicle Routing Problem". In LNCS 2279, pp. 11—20, Springer, April (2002).

Paessens, H.: The savings algorithm for the vehicle routing problem, Eur. J. Oper. Res. 34, 336-344, (1988).

5

Dynamic Vehicle Routing for Relief Logistics in Natural Disasters

Che-Fu Hsueh[1], Huey-Kuo Chen[2] and Huey-Wen Chou[2]
[1]Ching Yun University
[2]National Central University
Taiwan

1. Introduction

This chapter will introduce the dynamic vehicle routing problem for relief logistics in natural disasters such as earthquakes or typhoons. Natural disasters often cause huge fatalities and property damages. It is apparent that the coordinated and orderly delivery/pickup of available resources helps to mitigate property damages and save lives. Without performing quick and appropriate disaster relief logistics, the losses could be worsened. These relief resources, including food, water, medicine, and other equipment, may be misplaced at first due to the chaos and need to be rearranged in later time, according to the latest situation.

Disaster relief logistics using the vehicle routing approach has not attracted much attention and, hence, literature on this topic is relatively rare. Barbarosoglu et al. (2002) present a mathematical model for helicopter mission planning during a disaster relief operation. The decisions inherent in the problem decompose hierarchically into two sub-problems where tactical decisions are made at the top level, and the operation routing and loading decisions are made at the based level. Consistency between the decomposed problems is achieved with an iterative coordination procedure, which transfers anticipated information for the base level to improve the top-level decisions. The existence of conflicting objectives in this hierarchical structure requires the development of a multi-criteria analysis, and an iterative procedure is designed with top-level decision makers to assess the preference of alternative non-dominated solutions.

Mohaymany et al. (2003) propose a method to obtain the "emergency paths" that is based on two "life loss mitigation criteria". One is minimization of travel time between rescue and relief centers and the help-needing population, and the other is maximization of rescue and relief forces' service capability to all help-needing population. The main elements considered in the process of emergency paths selection include the highly populated areas and vulnerable and hazardous areas on the one hand, and rescue and relief centers on the other. The proposed path selection process is performed by using some developed computer programs in the GIS environment.

Özdamar et al. (2004) offer a planning model that is to be integrated into a natural disaster logistic decision support system. The model addresses the dynamic time-dependent transportation problem that needs to be solved repetitively at a given time interval during ongoing aid delivery. The model regenerates plans incorporating new requests for aid

materials, new supplies, and transportation means that become available during the current planning time horizon. The plan indicates the optimal mixed pickup and delivery schedules for vehicles within the considered planning time horizon, as well as the optimal quantities and types of loads picked up and delivered on these routes.

Thomas and White (2004) analyze the problem of constructing a minimum expected total cost route from an origin to a destination that anticipates and responds to service requests if they occur while the vehicle is en route. This problem is modeled as a Markov decision process and several structured results associate with the optimal expected cost-to-go function and an optimal policy for route construction are presented.

Pettit and Beresford (2005) present research that proposes a refined model for logistics requirements in emergency conditions, taking account of existing response models, both military and non-military. The composite model appears to be robust and workable in a range of geopolitical and operational circumstances.

Yi and Özdamar (2007) describes an integrated location-distribution model for coordinating logistics support and evacuation operations in disaster response activities. The logistics planning considered involves dispatching commodities (e.g., medical materials and personnel, specialized rescue equipment and rescue teams, food, etc.) to distribution centers in affected areas and evacuation and transfer of wounded people to emergency units. The proposed model is a mixed integer multi-commodity network flow model that treats vehicles as integer commodity flows rather than binary variables.

Gong et al. (2007) consider ambulance allocation and reallocation models for a post-disaster relief operation. They formulate a deterministic model that depicts how a casualty cluster grows after a disaster strikes and consider the objective of minimizing the makespan to determine allocation and reallocation of ambulances.

A real-time time-dependent vehicle routing problem with time windows has been formulated as a series of mixed integer programming models (Chen et al., 2005), which provides some basic knowledge of this chapter. A clear definition of "critical node" is proposed, which defines the scope of the remaining problem along the time-rolling horizon. The concept of critical nodes distinguishes the real-time vehicle routing problem with time windows (VRPTW) from the traditional VRPTW to a high degree. A heuristic comprising route construction and route improvement is proposed. Following the same line, Chang et al. (2003) further extended an earlier version of their work to accounts for simultaneous delivery/pickup demands while neglecting time-dependent assumption of travel times.

In summary, in the disaster relief operations, the logistic coordinator may not be aware of all information for route scheduling at the time when the routing process begins. Dynamic information, including delivery/pickup demands and travel times, may change after the initial routes have been constructed, and such kinds of information are known only in a real-time manner. When a new demand appears, the main task of the logistics coordination center is to include the new demand into the current routing schedule. In addition, if travel times have changed due to an unexpected incident, in order to fulfill time window constraints and achieve a lower cost objective, scheduled routes must be re-scheduled based on the positions of the vehicles. The logistics coordination center needs an answer urgently in order to respond in real time to both the real-time demands and travel times.

2. Model formulation for disaster relief operations

In this chapter, the dynamic vehicle routing problem for relief logistics in natural disasters (DVRP-RL) is described as follows. The demands for relief goods, including delivery or

pickup, may vary in real time and are probably unpredictable. The change of travel times may be predictable (e.g., recovery of a road, regulation of traffic, or recurrent traffic) or unpredictable, such as traffic incidents. The predictable travel time can be described as a function of "time of day" and the pattern of this function may change due to unpredictable incidents. At each demand site (or "node" in the following), several kinds of relief goods with different levels of emergency and deadlines are required and both delivery and pickup operations may be needed simultaneously. The excess relief goods or rescue equipment picked up at one site may be reused in any of the following sites of the same tour. For a quick response to future demands, a dispatched vehicle may not have to return to the depot unless it needs to be refilled with relief goods. It may wait at their last stop until they receive the next order from the logistics coordination center.

In order to capture the dynamic characteristics of the DVRP-RL, we adopt the concept in Chen et al. (2005) and formulate the DVRP-RL as a series of mixed integer programming models in the rolling time horizon. Each model represents a special VRPTW at a particular instant when travel times or demands have changed. Once a new demand appears or the travel time changes, the domain of the remaining problem should be redefined and the problem needs to be solved again. The concept of "critical node" is adopted to delineate the scope of the remaining problem along the rolling time horizon. A vehicle may depart from either the depot or the critical node and visit other unserviced nodes afterward. The goal of the model at any particular time instant τ is to find a set of minimum cost vehicle routes, which originate from the critical nodes or the depot, $N_c(\tau) \cup \{0\}$, and visit all other unserviced nodes, $N_u(\tau)$.

Figure 1 illustrates where the critical nodes is located. A critical node is defined as a node that is currently being serviced or to which a vehicle is heading. Once a vehicle has left a serviced node, the node will be removed from the planned route thereafter. Only unserviced nodes will be considered in the scheduled vehicle routes. Therefore, the critical node plays an important role in distinguishing between serviced nodes and unserviced nodes. Critical nodes need to be identified instantly when a real-time demand or travel time has changed so that the route can be reconstructed.

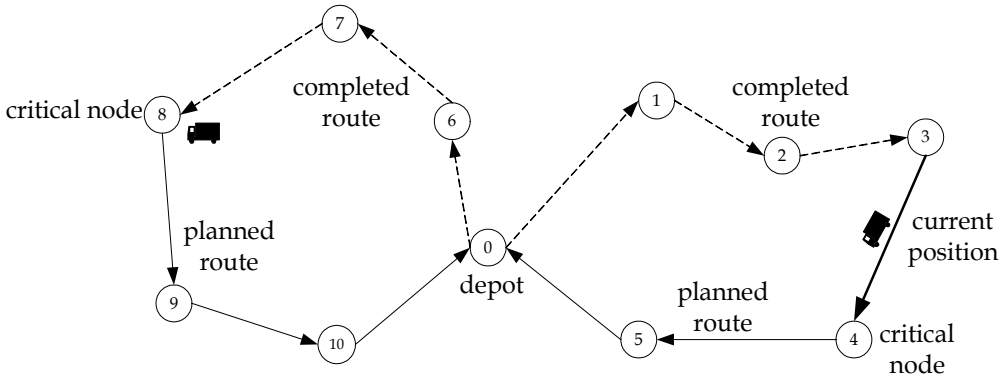


Fig. 1. Vehicle routes and critical nodes (Chen et al., 2005)

For ease of reference, notations are listed below:

Parameters and constants:

τ : time when real-time information (travel times or demands) occurs plus a small computation time

$c_{0j}(d_{0k})$: travel time between depot *0* and node *j* when vehicle *k* departs from the depot at time d_{0k}

$c_{ij}(d_i)$: travel time between nodes *i* and *j* at departure time d_i

C_k : full capacity of vehicle *k*

l_i : deadline at node *i*

q_{im} : delivery demand of commodity *m* at node *i*

$\hat{q}_{im}$: pickup demand of commodity *m* at node *i*

$\overline{Q}_{mk}$: total loads of commodity *m* for vehicle *k* when dispatching from the depot

$\overline{Q}_{mk}(\tau)$: remaining loads of commodity *m* for vehicle *k* at time τ

s_i : service time at node *i*

W_i : unit penalty of late arrival at node *i*

Sets:

{0} : depot

$N_c(\tau)$: set of critical nodes at time τ

$N_u(\tau)$: set of unassigned (or unserviced) nodes at time τ

$N_{cu}(\tau)$: set of critical nodes and unassigned nodes at time τ

$N_{u0}(\tau)$: set of the depot and unassigned nodes at time τ

$N_{cu0}(\tau)$: set of the depot, critical nodes and unassigned nodes at time τ

K : set of all vehicles, which is the union of two sets, i.e., in the depot and en route

$K_0(\tau)$: set of vehicles in the depot at time τ

M : set of all commodities needed in the disaster area

Variable:

a_i : time arriving at node *i*

d_i : time to depart node *i*

d_{0k} : time for vehicle *k* to depart from the depot

x_{ijk} : 1, if vehicle *k* departs node *i* toward node *j*; 0, otherwise

Q_{jmk} : remaining loads of commodity *m* when vehicle *k* arrives at node *j*

Whenever real-time demands have appeared or time-dependent travel times have changed, the submodel of DVRP-RL at this instant τ needs to be solved for an initial solution within a small computation time, which is very close to zero and will be neglected in the following. The submodel of DVRP-RL at time τ, denoted as VRP-RL, can be formulated as a mixed integer model, as follows:

$$\min_{\{x_{ijk}\}\in\Omega} z(\tau) = \sum_{j\in N_{u0}(\tau)} \sum_{k} \left(\sum_{i\in N_{cu}(\tau)} c_{ij}(d_i) x_{ijk} + c_{0j}(d_{0k}) x_{0jk} \right) + \sum_{i\in N_u(\tau)} W_i(\max(a_i - l_i, 0)) \qquad (1.)$$

where Ω is the feasible region represented by the following constraints.

Flow conservation constraints:

$$\sum_j \sum_k x_{ijk} = 1 \qquad \forall i \in N_{cu}(\tau) \tag{2.}$$

$$\sum_i \sum_k x_{ijk} = 1 \qquad \forall j \in N_u(\tau) \tag{3.}$$

$$\sum_i x_{ihk} - \sum_j x_{hjk} = 0 \qquad \forall\, h \in N_u(\tau), k \in K \tag{4.}$$

$$\sum_j x_{ij\bar{k}_i} = 1 \qquad \forall i \in N_c(\tau) \tag{5.}$$

$$\sum_j x_{0jk} \le 1 \qquad \forall k \in K_0(\tau) \tag{6.}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i \in N_{cu0}(\tau), j \in N_{u0}(\tau), k \in K \tag{7.}$$

Vehicle capacity constraints:

$$Q_{imk} \ge q_{im} \qquad if \quad \sum_j x_{ijk} = 1 \qquad \forall i \in N_u(\tau), m \in M, k \in K \tag{8.}$$

$$\sum_m (Q_{imk} - q_{im} + \hat{q}_{im}) \le C_k \qquad if \quad \sum_j x_{ijk} = 1 \qquad \forall i \in N_u(\tau), k \in K \tag{9.}$$

Definitional constraints:

$$a_j = d_i + c_{ij}(d_i) \qquad if \quad x_{ijk} = 1 \qquad \forall i \in N_{cu}(\tau), j \in N_u(\tau), k \in K \tag{10.}$$

$$a_j = d_{0k} + c_{0j}(d_{0k}) \qquad if \quad x_{0jk} = 1 \qquad \forall j \in N_u(\tau), k \in K \tag{11.}$$

$$d_i = \begin{cases} \max(\tau, a_i + s_i) & if \quad \sum_k x_{i0k} = 0 \\ \infty & if \quad \sum_k x_{i0k} = 1 \end{cases} \qquad \forall i \in N_{cu}(\tau) \tag{12.}$$

$$d_{0k} = \tau \qquad \forall k \in K_0(\tau) \tag{13.}$$

$$Q_{jmk} = Q_{imk} - q_{im} + \hat{q}_{im} \qquad if \quad x_{ijk} = 1 \qquad \forall i \in N_u(\tau), j \in N_u(\tau), m \in M, k \in K \tag{14.}$$

$$Q_{jmk} = \overline{Q}_{mk}(\tau) - q_{im} + \hat{q}_{im} \qquad if \quad x_{ijk} = 1 \qquad \forall i \in N_c(\tau), j \in N_u(\tau), m \in M, k \in K \tag{15.}$$

$$Q_{jmk} = \overline{Q}_{mk} \qquad if \quad x_{0jk} = 1 \qquad \forall j \in N_u(\tau), m \in M, k \in K \tag{16.}$$

The objective of the VRP-RL, shown in Eq. (1), is to minimize the total travel time of all tours and the total penalty due to late arrivals. The unit penalty W_i reflects the relative importance of node *i* for late arrival. Eq. (2) requires that only one vehicle can leave a critical node or unserviced node *i* once. Eq. (3) denotes that only one vehicle can arrive at unserviced node *j* once. Eq. (4) states that for each unserviced node *h*, the approaching vehicle must eventually leave this node. Eq. (5) requires that vehicle $\bar{k}_i$, which has arrived at or is approaching a critical node, must also leave this node once. Note that vehicle $\bar{k}_i$ is known at time τ. Eq. (6) requires that each vehicle can leave the depot once at most. Eq. (7) designates x_{ijk} as a 0-1 integer variable. x_{ijk} equals 1 if vehicle *k* departs node *i* toward node *j*. Otherwise, x_{ijk} equals 0.

Eq. (8) indicates that if node *i* is serviced by vehicle *k*, i.e., $\sum_j x_{ijk} = 1$, then the remaining load of commodity *m* of vehicle *k* at node *i* must be greater than or equal to the demand of commodity *m* at node *i*. Eq. (9) indicates that if $\sum_j x_{ijk} = 1$, then the updated load of vehicle k at node i after delivery and pickup must be less than or equal to the full capacity of vehicle *k*.

Eq. (10) and Eq. (11) define the arrival time at node *j*. Eq. (12) defines the departure time at node *i*. If $i \in N_u(\tau)$ then the departure time is set as $d_i = a_i + s_i$. On the other hand, if $i \in N_c(\tau)$, two scenarios must be further considered. (i) If node *i* is not the last node, i.e. $\sum_k x_{i0k} = 0$, then the departure time is set as $d_i = \max(\tau, a_i + s_i)$. The only situation that the departure time will take on the value of τ is when the vehicle that is waiting at the last node *i* receives a new order from the logistic coordination center and must depart for the next node immediately. (ii) If node *i* is the last node, i.e. $\sum_k x_{i0k} = 1$, then the departure time is set as $d_i = \infty$, referring to infinity. This logical expression is meant that vehicle *k* can stay at the last node *i* until the next order from the logistics coordination center has been received. For those vehicles that have to return to the depot to refill, pickup demands at the depot will be virtually created for them. Eq. (13) defines the departure time at the depot. Eq. (14) states for unserviced node *i* that if vehicle *k* is heading to node *j*, then the remaining load of commodity *m* of vehicle *k* at node *j* will be equal to the updated load of commodity *m* minus the delivery and plus the pickup at node *i*. Eq. (15) is identical to Eq. (14) except that the unserviced nodes are substituted by critical nodes. Eq. (16) defines the initial load of commodity *m* for the vehicle *k* dispatched from the depot.

Note that with departure time set as $d_i = \infty$ in Eq. (13), link $i \rightarrow 0$ will never be traversed in the solution at this instant, and hence link travel time between any node *i* and the depot, $c_{i0}(d_i)$, should not be included in the objective function. Consequently, model (1) is naturally modified as follows.

$$\min_{\{x_{ijk}\} \in \Omega} z(\tau) = \sum_{j \in N_u(\tau)} \sum_k \left(\sum_{i \in N_{cu}(\tau)} c_{ij}(d_i) x_{ijk} + c_{0j}(d_{0k}) x_{0jk} \right) + \sum_{i \in N_u(\tau)} W_i \left(\max(a_i - l_i, 0) \right) \qquad (17.)$$

3. Solution algorithm

We propose a two-phase heuristic comprised of routes construction and routes improvement for the DVRP-RL. For real-time operations, an anytime algorithm is desired. In other words, the solution procedure must have the ability to stop at any time and provide an acceptable solution. A unified solution procedure for the DVRP-RL is shown in Figure 2.

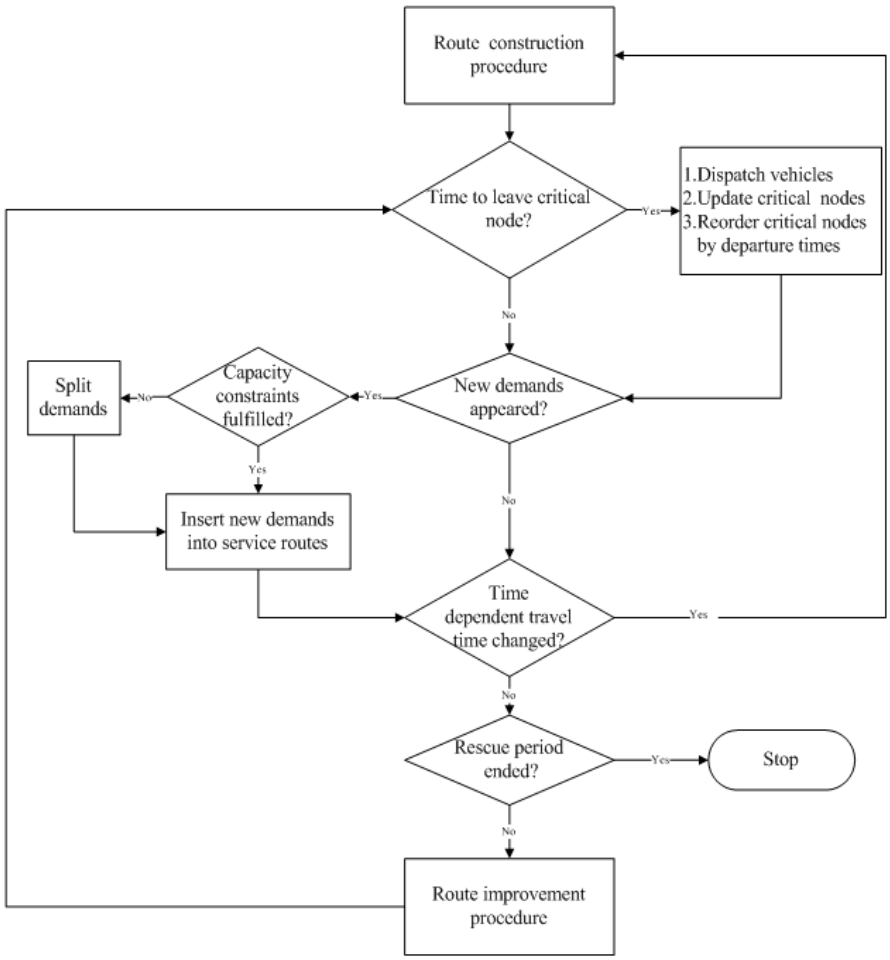


Fig. 2. Unified framework of solution procedure

The route construction and route improvement procedures are two major phases in the unified solution procedure. Between these two phases, we constantly check whether the following incidents happen.

i. Departure time of earliest critical node has arrived.
ii. New demand has appeared.
iii. Time-dependent travel time has changed.
iv. Disaster relief operation has ended.

The change of time-dependent travel times refers to the change of the travel time function, not merely current travel time itself. The operating maneuvers, such as dispatching en route

or on-call vehicles to the next demand site, reconstructing routes, improving the quality of the existing routes, and so forth, are repeatedly applied. Note that the computation time allowed for route construction and improvement is usually short due to the requirement of real-time response. It is necessary to adopt a fast solution algorithm for use. In our solution procedure, the insertion method is used for route construction, while the Or-opt node exchange algorithm is adopted for route improvement.

For the VRPTW, the insertion method has been proven effective in constructing static routes (Solomon, 1987). So we adopted the insertion method with modifications for the DVRP-RL in the phase of route construction, as follows.

Step 1: Input data

Input real-time demands of each commodity $\{q_{im}, \hat{q}_{im}\}$, deadlines $\{l_i\}$, service times $\{s_i\}$, time-dependent travel times $\{c_{ij}(d_i)\}$, and time instant τ.

Step 2: Identify the scope of the remaining problem

Classify nodes into critical nodes $N_c(\tau)$ and unassigned nodes $N_u(\tau)$.

Step 3: Find the inserting node with the smallest insertion cost

Find the inserting node (and its corresponding insertion position) that has the smallest insertion cost among all unassigned node $u \in N_u(\tau)$.

Step 4: Insert the selected node and update the relevant data

Update the system by inserting selected node *u* into the position with *smallest* accrued cost in an appropriate route and redefine relevant data, such as departure time and arrival time for each affected node. Once inserted, node *u* is then removed from set $N_u(\tau)$.

Step 5: Stopping check for assignment

If set $N_u(\tau)$ is empty, the phase of route construction terminates. Otherwise, compute the remaining delivery/pickup capacity. If no vehicles can fulfill capacity constraints for nodes in $N_u(\tau)$, split a large demand into several small parts for partial service. Go to Step 3.

After the routes are constructed, many existing route improvement methods can be applied, such as 2-opt and Or-opt, which may combine with other meta-heuristics.

Due to the fact that insufficient supplies in relief logistics is very common, the demand at one node can be split into several parts and each part corresponds to a "virtual" node with a different deadline and a different level of emergency. In this way, partial delivery is possible so that the limited relief can be distributed to more demand sites to ease the impact of the disaster, though the demands are not fully satisfied.

4. Computational results

The test problem set used in Chen et al. (2005) is adopted with additional columns added to accommodate real-time delivery/pickup demands of each commodity and the levels of emergency. We only perform the test of problem numbered R103 for illustration. It is assumed that two kinds of commodities are considered for delivery/pickup. The predictive traffic condition is characterized by a step-wise travel time function, which is exactly the same as the one used in Chen et al. (2005). During the disaster relief logistics, the number of unexpected road incidents/recoveries is set as four, at each of which a new travel time function will result. The whole vehicle capacity is set as 200 pallets for each vehicle, and the initial loads of a vehicle at the depot are 80% of the whole vehicle capacity. In other words, 80 pallets for each commodity are loaded when a vehicle is departing the depot. Other

information associated with each node, including the severity of emergency, which is further divided into three levels denoted as A, B, and C, is summarized in Table 1. Nodes in the class of "A" belong to the highest level of emergency and once its deadline is missed, the penalty is also the highest.

Appearing Time	Node no.	Level of emergency	Pickup commodity		Delivery commodity		Deadline
			Type	Quantity (pallets)	Type	Quantity (pallets)	
0	1	B	1	10	2	8	204
0	2	C	2	7	1	5	202
0	3	B	1	13	2	17	197
0	4	A	2	19	1	22	159
0	5	A	1	26	2	38	199
0	6	C	1	3	2	2	109
0	7	B	2	5	1	5	198
0	8	B	1	9	2	13	105
0	9	A	1	16	2	16	107
0	10	B	1	16	2	17	134
0	11	C	2	12	1	10	77
0	12	A	2	19	1	12	205
0	13	A	2	23	1	32	169
0	14	A	1	20	2	12	187
0	15	C	2	8	1	5	71
0	16	B	2	19	1	18	190
0	17	C	1	2	2	3	167
0	18	C	2	12	1	17	204
0	19	A	2	17	1	9	187
0	20	C	2	9	1	5	188
0	21	B	1	11	2	10	201
0	22	A	2	18	1	11	107
0	23	A	2	29	1	19	78
0	24	C	2	3	1	4	190
0	25	C	2	6	1	6	182
0	26	A	1	17	2	25	208
0	27	B	1	16	2	15	47
0	28	B	1	16	2	17	213
0	29	C	2	9	1	13	190
0	30	A	2	21	1	28	81
0	31	A	1	27	2	40	202
0	32	A	1	23	2	16	186
0	33	B	2	11	1	8	47
0	34	A	1	14	2	21	183
0	35	B	2	8	1	9	153
0	36	C	1	5	2	3	51
0	37	B	2	8	1	7	198

Appearing Time	Node no.	Level of emergency	Pickup commodity		Delivery commodity		Deadline
			Type	Quantity (pallets)	Type	Quantity (pallets)	
0	38	B	2	16	1	15	93
0	39	A	2	31	1	18	54
0	40	B	1	9	2	10	95
0	41	B	2	5	1	3	107
0	42	C	2	5	1	5	41
0	43	B	1	7	2	10	185
0	44	A	2	18	1	15	79
0	45	B	1	16	2	16	42
0	46	C	2	1	1	1	184
0	47	C	2	27	1	21	185
0	48	A	2	36	1	49	192
0	49	A	1	30	2	35	118
0	50	A	2	13	1	250	203
0	51	C	2	10	1	10	193
0	52	C	2	9	1	9	208
0	54	B	1	18	2	18	197
0	57	B	2	7	1	4	196
0	59	A	2	28	1	28	202
0	60	C	2	3	1	2	201
0	61	B	2	13	1	19	194
0	63	B	2	10	1	12	185
0	64	C	2	9	1	9	83
0	65	A	2	20	1	12	61
0	71	C	2	15	1	12	180
0	72	A	2	25	1	22	197
0	73	C	2	9	1	13	199
0	75	B	2	18	1	15	192
0	81	A	1	26	2	36	192
0	82	C	1	16	2	21	196
0	83	C	2	11	1	11	198
0	85	A	1	41	2	42	196
0	86	A	2	35	1	37	184
0	89	B	2	15	1	15	211
0	90	B	1	3	2	2	187
0	91	C	2	1	1	1	194
0	92	C	2	2	1	1	28
0	94	A	2	27	1	34	207
0	95	C	1	20	2	12	205
0	96	A	1	11	2	8	204
0	97	B	2	12	1	12	202
0	98	C	1	10	2	7	198

Appearing Time	Node no.	Level of emergency	Pickup commodity		Delivery commodity		Deadline
			Type	Quantity (pallets)	Type	Quantity (pallets)	
10	67	A	1	25	2	13	93
10	78	C	1	3	2	3	106
11	62	B	2	19	1	18	68
12	69	C	1	6	2	4	60
19	79	A	2	23	1	31	102
19	80	C	1	6	2	8	192
23	87	B	1	26	2	31	103
28	76	B	1	13	2	7	83
31	68	A	1	36	2	33	152
32	88	C	2	9	1	8	84
33	100	B	2	17	1	19	195
35	99	A	1	9	2	6	93
38	58	C	2	18	1	24	210
44	53	B	1	14	2	13	105
48	66	A	2	25	1	28	137
50	77	B	1	14	2	9	189
55	84	C	1	7	2	7	111
62	56	B	1	6	2	5	140
88	55	C	1	2	2	1	146
91	74	B	2	8	1	9	159
125	93	A	2	22	1	24	198
136	70	C	1	5	2	7	192

Table 1. Delivery/pickup information at demand sites

In order to show the capability for partial delivery, we set the delivery demand at node 50 equal to 250 pallets, which is greater than the vehicle capacity of 200 pallets. It is obviously that only partial demand, preset as 50 pallets, can be satisfied by the first incoming vehicle. The unsatisfied delivery demand, say 200 pallets, will stand at the same node with the same deadline and wait for service during the solution process. If the unsatisfied delivery demand can still not be satisfied by the second incoming vehicle, the partial delivery will be performed again. This process will be repeated until the entire delivery demand at this node has been totally satisfied.

The testing result shows that acceptable initial routes responding to the change of real time information were obtained in a very short time – less than one second for a problem with 100 nodes. The quality of the solution is further improved after the initial routes were obtained. This computational efficiency implies that the proposed model and associated algorithm are very suitable for real-time disaster relief logistics because of its quick response requirement.

The testing result is shown in Table 2, in which the number of vehicles is limited to 10. It shows that node 50 is serviced by three different vehicles – i.e., vehicles 1, 2, and 6 – and their respective delivery quantities are 50, 100, 100 pallets, implying that partial delivery is actually made. It is also found that 28 out of 100 nodes cannot be serviced by their deadlines; however, the routing schedule persists with late arrivals. In fact, late arrivals can be avoided

by increasing the number of vehicles, as in Figure 3. The number of late arrivals decreases apparently as the number of vehicles increases. All nodes can be serviced by the deadlines if the number of vehicles increases to 17.

Veh. No.	Route	Commodity 1 (pallets)			Commodity 2 (pallets)		
		$\overline{Q}_{mk}$	$\sum_i q_{im}$	$\sum_i \hat{q}_{im}$	$\overline{Q}_{mk}$	$\sum_i q_{im}$	$\sum_i \hat{q}_{im}$
1.	0→27→50→30→20→10→63→ 90→11→64→70	80	114	40	80	41	61
2.	0→28→12→76→77→ 3→79→ 78→68→50→26→80→29	80	156	118	80	119	51
3.	0→42→87→92→99→94→37→ 98→100→91→85→93→59→17	80	119	88	80	89	110
4.	0→45→8→83→60→5→84→ 61→16→86→96→18	80	104	69	80	82	93
5.	0→40→21→72→23→22→75→ 74→53→58→2	80	105	34	80	33	123
6.	0→33→51→9→69→1→50→ 34→81→50→24	80	122	72	80	85	37
7.	0→65→35→71→66→32→19→ 82→47→46	80	92	39	80	37	113
8.	0→44→38→14→43→6→13→ 95→97→57→15	80	83	50	80	36	84
9.	0→ 4→39→67→56→41→55→ 25→54→73	80	62	51	80	37	70
10.	0→36→49→62→88→31→48→ 7→89→52	80	104	62	80	78	93

$\overline{Q}_{mk}$: initial loads on the vehicle

$\sum_i q_{im}$: total amount of delivered commodity m on the vehicle

$\sum_i \hat{q}_{im}$: total amount of pick-up commodity m on the vehicle

Table 2. Computational results (No. of vehicles = 10)

To show the importance of real-time information, we created a benchmark solution that does not take into account the change of real-time travel times. In other words, the scheduled routes in the benchmark solution remain unchanged regardless of how the real-time travel times vary – that is, without considering real-time information. The number of nodes encountering late arrival is 39 in the benchmark solution as opposed to 28 nodes in the original test solution. In addition, the total penalty of late arrival obtained in the benchmark problem is about 1.9 times as high as that of the original test problem.

It is also worth mentioning that some relief goods or rescue equipments at a demand site may be no longer desired and, therefore, can be picked up for use in the following nodes of the same route. For example, in Table 2 the total quantity of delivery for commodity 1 by vehicle 1 is 114 pallets, which is greater than its initial loads, 80 pallets, implying that at least some delivery demands are fulfilled by the pickup commodity from previous nodes of the same route.

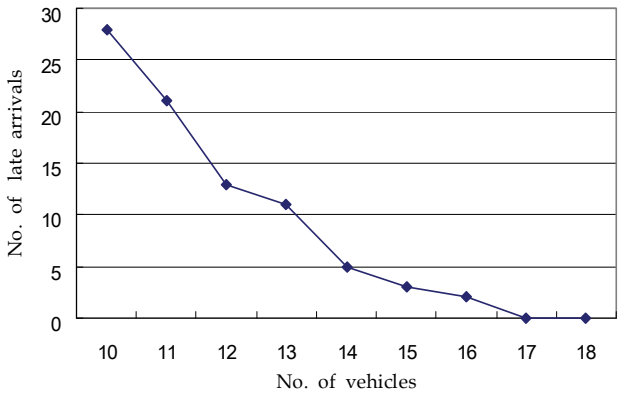


Fig. 3. Number of late arrivals under different fleet size

5. Conclusion

In the operations of relief logistics in natural disasters, there are several features that are different from the traditional VRPTW, such as unexpected road damage and recovery, real-time information and response, partial delivery, re-allocation of resource, different level of emergency, and so on. The model and algorithm of DVRP-RL are proposed to meet these requirements and provide an efficient solution for relief logistics. Some contributions are summarized as follows:

1. Real-time information such as road incidents, road/bridge damage recoveries, or delivery/pickup requests can be considered from time to time to regenerate a better route schedule.
2. When a visited demand site needs further disaster relief, it will be regarded as a new demand and taken back to the unserviced node list. This situation is quite common in a disaster relief operation if the damage resulting from the natural disaster continues.
3. Delivery and pickup demands are considered simultaneously. In addition, the excess relief goods or rescue equipment picked up at one site can be reused in the following sites of the same route. In reality, the coordination and reallocation of relief resource between different demand sites is essential in disaster relief operations.
4. Partial or full delivery/pickup operations, which are critical in the disaster relief logistics, can be accommodated in the solution procedure.
5. Relative importance of a demand node for late arrival is reflected in the corresponding unit penalties due to missing the deadline. When late arrivals are inevitable, the service sequence among those nodes can be determined based on the magnitude of the corresponding unit penalties.
6. Without having to return to the depot, a vehicle can stay at the last node, waiting for the next order from the logistics coordination center, and hence, has more flexibility for disaster relief operations in natural disasters. This treatment is useful especially when the depot is distant from the disaster area.
7. Supply shortages were implicitly considered by setting the deadline and classifying all demand nodes into several levels of emergency.

Two suggestions are made in the following for future research.

1. The optimal initial loads of vehicles need further study, especially in a real-time situation.
2. Anticipatory routing, which analyzes the problem of constructing a minimum expected cost route from an origin to a destination, anticipates and then responds to service requests.

8. References

Barbarosoglu, G., Ozdamar, L. & Cevik, A. (2002). An interactive approach for hierarchical analysis of helicopter logistics in disaster relief operations, *European Journal of Operational Research,* Vol.140, 118-133.

Chang, M.S., Chen, S.R. & Hsueh, C.F. (2003). Real-time vehicle routing problem with time windows and simultaneous delivery/pickup demands, *Journal of the Eastern Asia Society for Transportation Studies,* Vol.5, 2273-2286.

Chen, H.K., Hsueh, C.F. & Chang, M.S. (2005). The real-time time-dependent vehicle routing problem, *Transportation Research Part E,* Vol.42, No.5, 383-408.

Gong, Q. & Batta, R. (2007) Allocation and reallocation of ambulances to casualty clusters in a disaster relief operation, *IIE Transactions (Institute of Industrial Engineers)* Vol.39, N0.1, 27-39.

Mohaymany, A. S., Hosseini, M. & Habibi, H. M. (2003). Obtaining the emergency transportation network for rescue and relief activities in large cities based on the life loss mitigation criteria. *Technical Council on Lifeline Earthquake Engineering Monograph,* Vol.25, 231-240.

Özdamar, L., Ekinci, E. & Küçükyazaci, B. (2004). Emergency logistics planning in natural disasters, *Annals of Operations Research,* Vol.129, 217-245.

Pettit, S. J. & Beresford, K. C. (2005). Emergency relief logistics: An evaluation of military, non-military and composite response models, *International Journal of Logistics: Research and Applications,* Vol.8, No.4, 313-331.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time windows constraints, *Operations Research,* Vol.35, 254-265.

Thomas, B. W. & White III, C. C. (2004). Anticipatory route selection, *Transportation Science,* Vol.38, No.4, 473-487.

Yi, W. & Özdamar, L. (2007). A dynamic logistics coordination model for evacuation and support in disaster response activities, *European Journal of Operational Research,* Vol.179, No.3, 1177-1193

6

Cumulative Vehicle Routing Problems[(1)]

İmdat Kara[1], Bahar Yetiş Kara[2] and M. Kadri Yetiş[3]
[1]*Başkent University, Department of Industrial Engineering,*
[2]*Bilkent University, Department of Industrial Engineering,*
[3]*Havelsan A.Ş.7th km on Eskişehir Road,*
Ankara,
Turkey

1. Introduction

The problems of finding optimal routes for vehicles from one or several depots to a set of locations/customers are known as Vehicle Routing Problems (VRPs) and have many practical applications, especially in transportation and distribution logistics. An extensive literature exists on these problems and their variations (e.g. Bodin(1990), Laporte(1992), Laporte & Osman(1995), Ball et al.(1995), Toth & Vigo (2002a)).

The Capacitated Vehicle Routing Problem (CVRP) is defined on a graph G = (V, A) where V={0,1,2, …, *n*} is the set of nodes (vertices), 0 is the *depot (origin, home city)*, and the remaining nodes are *customers*. The set A = {(*i*, *j*): *i*, *j* ∈ V, *i* ≠ *j*} is an arc (or edge) set. Each customer $i \in V \setminus \{0\}$ is associated with a positive integer demand q_i and each arc (*i*, *j*) is associated a travel cost c_{ij} (which may be symmetric, asymmetric, deterministic, random, etc.). There are *m* vehicles with identical capacity *Q*. The CVRP consists of determining a set of *m* vehicle routes satisfying the following conditions:

- Each route starts and ends at the depot,
- Each customer is visited by exactly one route,
- The total demand of each route does not exceed the vehicle capacity *Q*,
- The total "cost" of all routes is minimized.

The CVRP has been studied extensively in the literature (for recent publications, see e.g. Achutan et al. (1996), Toth & Vigo (2002b), Ralphs et al. (2003), Baldacci et al. (2004), Kara et al. (2004), Letchford & Salazar-Gonzalez (2006), Yaman (2006)). CVRP was first defined by Dantzig and Ramser in 1959. In that study, the authors used distance as a surrogate for the cost function. Since then, the cost of traveling from node *i* to node *j*, i.e., c_{ij}, has usually been taken as the distance between those nodes.

The real cost of a vehicle traveling between two nodes depends on many variables: the load of the vehicle, fuel consumption per mile (kilometer), fuel price, time spent or distance traveled up to a given node, depreciation of the tires and the vehicle, maintenance, driver wages, time spent in visiting all customers, total distance traveled, etc. (Baldacci et al. (2004), Toth & Vigo (2002a), Desrochers et al. (1990)). Most of the attributes are actually distance or

[1] A preliminary version of this paper has appeared in the proceedings of the First International Conference on Combinatorial Optimization and Applications, COCOCA 2007, Xi'an, China, (Kara et al., 2007).

time based and can be approximated by the distance. However, some variables cannot be represented by the distance between nodes. Examples of such variables are vehicle load, fuel consumption per mile (kilometer), fuel price, time spent up to a given node. Most of these types of variables may be represented as a function of the flow on the corresponding arc (load or weight of the vehicle, number of items on the vehicle, the order in the tour of starting and/or ending node of the arc and etc.). Thus, for some cases, in addition to the distance traveled, we need to include flow on the related arc as another indicator of the cost. In this study, we propose a cost function which is defined as a product of the distance traveled and the flow on that arc. To the best of our knowledge, the vehicle routing literature has not previously included such a definition of cost, which is the main motivation of this research.

In the VRP, vehicles collect and/or deliver the items and/or goods from/to each customer on the route. Thus, the flows on the arcs change throughout the tour. They show an increasing step function in the case of collection and a decreasing step function in the case of delivery. So, flows cumulate or diminish along the tour. For this reason, we call a CVRP with flow based cost function as the Cumulative Vehicle Routing Problem, abbreviated as CumVRP.

The main contribution of this paper may be summarized as:

- Define a new cost function for vehicle routing problems as a multiple of length of the arc traveled and the flow on this arc. Name this problem as Cumulative Vehicle Routing Problem (CumVRP).
- Present polynomial size integer programming formulations for CumVRP for collection and delivery cases.
- Show the relationship between the proposed CumVRP with the m-Traveling Repairman and related problems .
- Illustrate the use of CumVRP in the real life situations such as energy minimizing VRP and school-bus routing problems.

We provide problem identification and integer programming formulations of the CumVRP for both collection and delivery cases in Section 2. In Section 3, we show that the proposed problem is a generalization of the m-Traveling Repairmen Problem, and so is relevant to minimum latency and its variations mentioned in the literature. Two additional applications of the flow-based cost function, namely the Energy-Minimizing VRP and the Average-Distance Minimizing School Bus-Routing Problem, are also illustrated in the same section. The proposed models are tested and illustrated by real life data from Turkey and the results are given in Section 4. Concluding remarks are in Section 5.

2. Formulations of the CumVRP

In this section, details of the Cumulative Vehicle Routing Problem are outlined and then integer linear programming formulations are presented.

2.1 Problem identification

Consider a vehicle routing problem defined over a network $G = (V, A)$ where $V=\{0,1,2, \ldots, n\}$ is the node set, 0 is the depot and $A = \{(i, j): i, j \in V, i \neq j\}$ is the set of arcs, and, components are given as:

Parameters:

d_{ij} is the distance from node i to node j.

q_i is the nonnegative weight (e.g. demand or supply) of node i.

m is the number of identical vehicles.

Q_0 is the initial value of flow from the origin to the first node of the tour in the case of collection, or the final value of flow from the last node of the tour to the origin in the case of the delivery, (e.g. tare of the truck in the case of carrying goods).
M represents the flow capacity of the arcs of the network (maximal value of the flow on any arc of the network, for example, capacity plus tare of the trucks in the case of carrying goods).

Decision Variables:

x_{ij} = 1 if the arc (i, j) is on the tour of a vehicle, and zero otherwise;
y_{ij} is the flow on the arc (i, j) if the vehicle (traveler) goes from i to j, and zero otherwise.

Cost:

The cost of traversing an arc (i , j), c_{ij}, is defined as the product of the distance of the arc (i , j) and flow on this arc.
With those given above, we define Cumulative Vehicle Routing Problem (CumVRP) as:

- Each node (customer) is served exactly by one vehicle.
- Each route starts and ends at the depot.
- For each tour, the flow on the arcs cumulate as much as preceding node's supply in the case of collection or diminish as much as preceding node's demand in the case of delivery.
- The flow on any arc of each tour doesn't exceed the flow capacity of the arcs.
- The objective is to find a set of m vehicle routes of minimum total cost where the cost is defined as the product of the distance of the arc (i , j) and flow on this arc.

Definition of the y_{ij}'s is the core of this approach. The flow on the first arc of any tour must take a predetermined value and then must always increase (or decrease) by q_i units just after node i. In the case of collection, the flow variable shows an increasing step function; for delivery, it shows a decreasing step function. Therefore a model constructed for collection case may not be suitable for the delivery case. The following observation states the relationship between them.

Observation 1: When the distance matrix is symmetric, the optimal route of the delivery (collection) case equals the optimal route of the collection (delivery) case traversed in the reverse order.

Proof: Consider a route which consist of k nodes: n_0- n_1-n_2-…-n_k-n_0, where n_0 is the depot. For the collection case, the cost of this tour is:

$$Q_0 d_{01} + \sum_{j=1}^{k-1}\left(Q_0 + \sum_{i=1}^{j} q_i\right) d_{j,j+1} + \left(Q_0 + \sum_{i=1}^{k} q_i\right) d_{k0} \tag{1}$$

For the delivery case, the cost of the reverse route n_0- n_k-n_{k-1}-…-n_1-n_0 is:

$$\left(Q_0 + \sum_{i=1}^{k} q_i\right) d_{0k} + \sum_{j=1}^{k-1}\left(Q_0 + \sum_{i=1}^{j} q_i\right) d_{j+1,j} + \; Q_0 d_{10} \tag{2}$$

Observe that (1) and (2) are the same for symmetric $D=[d_{ij}]$ matrices.□

2.2 Mathematical models

For the symmetric-distance case, one does not need to differentiate between collection and delivery since the solution of one will determine the solution of the other. For the case of an

asymmetric distance matrix, due to the structure of the problem, we present decision models for collection and delivery cases, separately. The model for the collection case is:

$$F_1: \ Min \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} y_{ij} \tag{3}$$

s.t.

$$\sum_{i=1}^{n} x_{0i} = m \tag{4}$$

$$\sum_{i=1}^{n} x_{i0} = m \tag{5}$$

$$\sum_{i=0}^{n} x_{ij} = 1 \tag{6}$$

$$\sum_{j=0}^{n} x_{ij} = 1 \tag{7}$$

$$\sum_{\substack{j=0 \\ j \neq i}}^{n} y_{ij} - \sum_{\substack{j=0 \\ j \neq i}}^{n} y_{ji} = q_i \qquad i = 1, 2, \ldots, n \tag{8}$$

$$y_{0i} = Q_0 \, x_{0i} \quad i = 1, 2, \ldots, n \tag{9}$$

$$y_{ij} \leq (M - q_j) x_{ij} \qquad (i, j) \in A \tag{10}$$

$$y_{ij} \geq (Q_0 + q_i) x_{ij} \quad \forall (i, j) \in A \tag{11}$$

$$x_{ij} = 0 \ or \ 1, \ (i, j) \in A \tag{12}$$

Where $q_0 = 0$.

The objective function given in (3) gives the proposed cost function. Constraints (4) and (5) ensure that *m* vehicles are used. Taking "≤" instead of " = " in these relation is also possible when one imposes to use at most m vehicle. Constraints (6) and (7) are the degree constraints for each node, together with (4) and (5), they are called assignment constraints of the formulation. Constraint (8) is the classical conservation of flow equation balancing inflow and outflow of each node, they guarantee that, flow variables of each tour perform an increasing step function. Those constraints also prohibit any illegal subtour. Constraint (9) initialize the flow on the first arc of each route, cost structure of the problem necessitates such an initialization. Constraints (10) take care of the capacity restrictions and forces y_{ij} to zero when the arc (i,j) is not on any route, and constraint (11) produce lower bounds for the flow on any arc. Integrality constraints are given in (12). We do not need nonneqativity constraints for y_{ij}'s since we have constraints given in (11).

Let us call constraints (9), (10) and (11) as the *bounding constraints* of the formulation . Validity of these bounding constraints is shown in proposition 1 below.

Proposition 1: In the case of collection, the constraints given in (9), (10) and (11) are valid for CumVRP.

Proof: As it is explained before, we need initialization value of y_{ij}'s for each tour that constraints (9) do it, otherwise y_{ij}'s may not be actual flow on the arcs. Constraints (11) is valid since going from *i* to *j* the flow must be at least the initial value plus the weight of the node *i* (unless node *i* is the depot, in which case $q_0 = 0$). Similarly, since the vehicle is destined for node *j*, it will also collect the weight at node *j* (unless *j* is the depot). In that case, the flow on the arc upon arriving at node *j* should be enough to take the weight of node *j*, i.e., $y_{ij} + q_j x_{ij} \leq M x_{ij}$, which produce constraints (10).□

Similar constraints for classical CVRP may be seen in (Gouveia (1995), Baldacci et al.(2004), Letchford & Salazar-Gonzalez (2006), Yaman (2006)).

Due to Observation 1, the delivery problem for the symmetric case need not be discussed. For the asymmetric case, the delivery problem will be modeled by replacing constraints (8) ,(9), (10) and (11) with the following given below.

$$\sum_{\substack{j=0 \\ j\neq i}}^{n} y_{ji} - \sum_{\substack{j=0 \\ j\neq i}}^{n} y_{ij} = q_i \quad \text{for } i=1,2,\ldots,n \tag{13}$$

$$y_{i0} = Q_0 x_{i0} \text{ for } i=1,2,\ldots,n \tag{14}$$

$$y_{ij} \leq (M - q_i) x_{ij} \quad \forall (i,j) \in A \tag{15}$$

$$y_{ij} \geq (Q_0 + q_j) x_{ij} \quad \forall (i,j) \in A \tag{16}$$

Thus the model for the delivery case is:

$$F_2: \quad Min \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} y_{ij}$$

s.t. (4)-(7),(12) - (16).

where $q_0=0$.

Both of the proposed models have n^2+n binary and n^2+n continuous variables, and $2n^2+6n+2$ constraints, i.e., proposed formulations contain $O(n^2)$ binary variables and $O(n^2)$ constraints.

3. Applications of the CumVRP

In this section, we show that the special case of the CumVRP turns out some routing problems, namely, Minimum Latency Problem, m-Traveling Repairman Problem, Energy Minimizing Vehicle Routing Problem and School-bus Routing Problem.

3.1 Relevance to minimum latency and related problems

The *Time-Dependent Traveling Salesman Problem* (TDTSP) is a generalization of the standard Traveling Salesman Problem in which the cost of traveling from one node to another depends not only on the two locations, but also on their positions in the tour (Picard &

Queyranne (1978), Lucena (1990), Gouviea & VoB (1995)). When the objective of the TDTSP is to minimize the sum of distances traveled from the depot to all nodes, the problem is known as the *Traveling Salesman Problem with Cumulative Cost* or the *Cumulative Traveling Salesman Problem* (CTSP), as defined by Bianco et al.(1993). The CTSP is exemplified by pizza delivery since the time it takes the pizza to reach the customer is determined by the total travel time from the depot.

Latency of a node is defined as the total distance traveled up to that node. The *Minimum Latency Problem* is to find a tour starting at a depot and visiting all nodes in such a way that the total latency is minimized (Archer et al. (2003), Blum et al. (1994)). This problem is also known as the *Delivery Man Problem* (Fischetti et al. (1993)) or the *Traveling Repairman Problem* (Jothi & Raghavachari (2007)).

We conclude therefore that, the Cumulative Traveling Salesman Problem, the Minimum Latency Problem and the Traveling Repairman Problem are all same with respect to the structure of their objective functions.

The *Multiple Traveling Repairman Problem* (mTRP) is a generalization of the Minimum Latency Problem (hence repairman) and finds m tours, each starting at the depot and covering all the nodes while minimizing total latency (Jothi & Raghavachari 2007).

We now investigate the relations between the CumVRP and the mTRP.

Lemma 1: mTRP is a special case of the delivery formulation of CumVRP, where Q_0 =1 and q_i =1 for all i=1, 2, …, n and M = n-m+2.

Proof: Consider delivery formulation of CumVRP. Define y_{ij} as the number of remaining arcs on the tour from the node i to the origin if traveler goes from i to j, zero otherwise. Letn Q_0 =1, and q_i =1 for all i=1,2,…,n. For each tour, the corresponding y_{ij}'s shows a descending ordered integer sequences ending with 1. For such a case, consider a route composed of k intermediate nodes as n_0- n_1-n_2-…-n_k-n_0, where n_0 is the depot. Let d_{ij} denote the distance from n_i to n_j of this route and y_{ij} are the corresponding flow variables. Since there are (k+1) nodes in this tour, in order to end with value 1, the starting value of y_{ij} must be (k+1). So the part of the objective function of the CumVRP model corresponding to the this route is,

$$(k+1)d_{01}+kd_{12}+(k-1)d_{23}+\ldots\ldots+d_{k0}, \tag{17}$$

which can be rewritten as,

$$d_{01} + (d_{01} +d_{12})+ (d_{01} +d_{12} +d_{23})+ \ldots..+ (d_{01} +d_{12} +d_{23}+\ldots+ d_{k0}). \tag{18}$$

As can be seen, expression (18) is the sum of the latencies of the nodes on this tour. Thus, the CumVRP formulation for this delivery problem involves finding m tours, each starting at the depot, such that the total latency up to each node is minimized; this is nothing but the mTRP problem. Hence, mTRP is a special case of the delivery formulation of the CumVRP.□

For this special case, m-1 traveler (repairman) may visit only one node and turn back to the depot. So maximal intermediate nodes on a tour will be n-(m-1) = n-m+1, which implies that, the value of the M in the formulation, i.e., the maximal number of arc on a tour, must be taken as n-m+1. Using Lemma 1, letting q_i =1 for all i, Q_0 =1 and M=n-m+1 in formulation F2, we produce a formulation for mTRP and so we propose an integer programming formulation for mTRP with $O(n^2)$ binary variables and $O(n^2)$ constraints as follows:

$$F_3: \ Min \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} y_{ij} \tag{3}$$

s.t. (4)-(7) ,(12) and

$$\sum_{\substack{j=0 \\ j \neq i}}^{n} y_{ji} - \sum_{\substack{j=0 \\ j \neq i}}^{n} y_{ij} = 1 \text{ for } i=1,2,\ldots,n \tag{19}$$

$$y_{i0} = x_{i0} \text{ for } i=1,2,\ldots,n \tag{20}$$

$$y_{ij} \leq (n-m)x_{ij} \ \ i \neq 0, \ \forall (i,j) \in A \tag{21}$$

$$y_{ij} \geq 2x_{ij} \ \ j \neq 0, \forall (i,j) \in A \tag{22}$$

In this formulation, constraints (19)-(22) play same role as the corresponding constraints do in F2In the formulation F3, if we take m=1, i.e., one traveler, we get an integer linear programming formulation for Minimum Latency (Traveling Repairman and Cumulative Traveling Salesman) Problem. Consequently, CumVRP produces a unified formulation for these special routing problems also.

It has been shown in the literature that the CTSP and related problems including the mTRP are NP-hard (Tsitsiklis (1992), Archer et al (2003)). Due to the Lemma 1, we conclude that CumCVRP is NP-hard.

For some cases of the mTRP (e.g. delivering pizza), the distance (or traveling time) of the last arc on the tour may not be considered in the objective function.

3.2 Energy minimizing vehicle routing problem [Kara et al., (2007)]

For vehicle routing problems where vehicles carry goods from an origin (center, factory and/or warehouse) to the customer, or from the customer to the origin, the traveling cost between two nodes can be written as,

Cost = f(load, distance traveled, others)

where f(.) is any function. We derive a cost function that mainly focuses on the total energy consumption of the vehicles. Recall from mechanics that,

Work = force * distance

In the VRP, the movement of the vehicles can be considered as an impending motion where the force causing the movement is equal to the friction force (see for example Walker (2000)). Remember also that,

Friction force = Coefficient of friction * weight.

Thus, we have

Work = Friction force * distance.

Work = Coefficient of friction * weight * distance

The coefficient of friction can be considered as constant on roads of the same type. Then, the work done by a vehicle over a link (*i, j*) will be:

Work = weight of the vehicle (over link (i, j)) * distance (of link (i, j)).

Since work is energy, minimizing the total work done is equivalent to minimizing the total energy used (at least in terms of fuel consumption). Obviously, the weight of the vehicle equals the weight of the empty vehicle (tare) plus the load of the vehicle. Thus, if one wants to minimize the work done by each vehicle, or to minimize the energy used, one needs to use the cost as,

$$\text{Cost of } (i, j) = [\text{Load of the vehicle over } (i, j) + \text{Tare}] * \text{distance of } (i, j), \quad (23)$$

There seems to be no such definition and objective cost function in the vehicle routing literature. However, there are references on the Internet (such as the Goodyear website) indicating that fuel consumption changes with vehicle load. The figure-1 below depicts the miles per gallon as a function of the total load.

Clearly, miles per gallon decrease with increased vehicle weight. Thus for a VRP in which goods are carried and fuel prices are relatively more important than the drivers' wages, considering the load of the vehicle as well as the distances will produce a more realistic cost of traveling from one customer to another. This analysis shows that for such VRP's we may define a more realistic cost of traveling from one customer to another by considering the load of the vehicle as well as the distances. We refer the VRP in which cost is defined as in expression (23) as the *Energy Minimizing Vehicle Routing Problem*, this being a special case of the CumVRP as shown below.

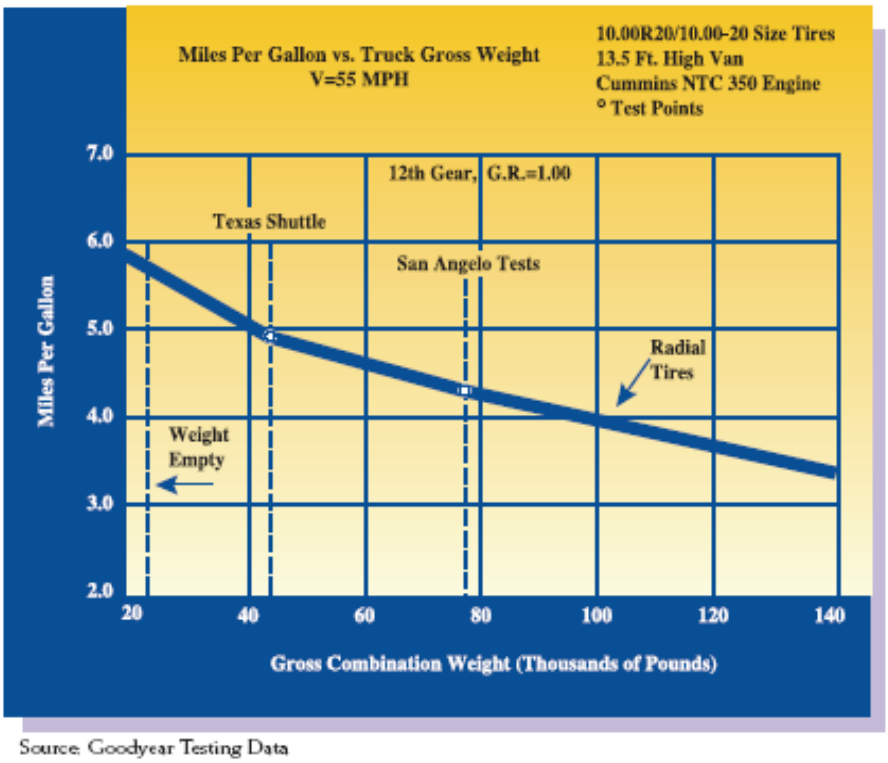


Figure 1. Miles per Gallon versus vehicle weight (http://www.goodyear.com/truck/pdf/commercialtiresystems/FuelEcon.pdf).

In CumVRP, let us define yij as the total weight of the truck while traversing the arc (i, j); qi 's are supply (collection) or demand (delivery) of ith customer; Q0 is the tare of the truck, Q is the capacity of each truck and M = Qo+Q. With these definitions, substituting Qo+Q instead of M in F1 and F2 , we get integer programming formulations of the collection and delivery cases of the Energy Minimizing Vehicle Routing Problem, respectively.

3.3 School-bus routing problem

School-bus routing is a primary application of the VRP; the question is how to transport students to and from school in an optimal way (Bodin (1990); Corberan et al. (2002)). The children are assigned to bus stops and a sequence of individual stops will form a bus route. In the morning, the buses pick up the students from bus stops and take them to the school; the procedure is reversed in the afternoon (Bodin (1990); Swersey & Ballard (1984)). Each school bus routing problem has different objectives and/or constraints. Bus capacity, the maximum number of stops per bus, the maximum length (or duration) of each tour and student riding time are the most frequently encountered additional requirements. There are different objectives, such as: minimizing transportation cost, minimizing transportation time, minimizing the time that a student spends on the bus, minimizing the number of buses required, minimizing fleet travel time, and balancing bus loads and route lengths (Corberan et al.(2002); Li & Fu (2002)).

For a school-bus routing problem, suppose the decision maker wants to evaluate alternative routes with respect to average distance traveled (or average time spent) per student. We could find no such objective in the literature. Let us define this problem as the *Average-Distance Minimizing School-Bus Routing Problem*. The CumVRP formulation can be used to solve such a school-bus routing problem.

Because of the asymmetric nature of road traffic at different times of a day, the collection and delivery cases in school-bus routing problems must be handled separately. Let us define V= {0, 1, 2… n} as a set of nodes (vertices), where {0} is the school and the remaining nodes are bus stops. Let the set A, the distances between two nodes d_{ij} and the binary variables x_{ij} be defined as in Section 2.1. Define y_{ij} as the number of students in the bus while it is traveling on the arc (i, j) if the bus passes from bus stop i to bus stop j, zero otherwise. We assume that the buses start their tour in the morning from the parking place and end at the school. In the afternoon, the tours begin from the school and ends at the parking place. For simplicity, we assume that the parking place is at the school. In computing the average distance traveled per student, the distance (or time) from the parking place to the first pick-up point in the collection case, and the distance from the last stop to the parking place in the delivery case, must not be considered; therefore Q_0 is equal to zero in both cases.

Consider a school bus problem where Cap denotes the capacity of the bus and q_i denotes the number of students boarding or disembarking at stop i. Then any average-distance minimizing school-bus routing problem is equivalent to CumVRP where Q_0=0 and M = Cap. So then with these parameters, F1 will be a formulation for morning tours (collection) and F2 will be a formulation for afternoon tours (delivery).

If there are other restrictions besides bus capacity, these constraints can easily be incorporated into the model. If the objective is defined as minimizing the average time spent per student, it is only necessary to measure the time needed to travel between the nodes of the arc set A and take the d_{ij}'s as the time parameters.

4. Illustrative examples and computational analsis

In this section, we conduct some numerical examples of CumVRP formulation focusing on the collection case of the Energy Minimizing VRP.

We solve the instances via CPLEX 8.1. on an Intel Pentium III 1400 MHz computer. We want to test the effect of the new objective function on the optimal routes (i.e. distance-based routes versus energy-based routes).

The Energy Minimizing model of CumVRP is tested for realistic instances by using the data from the Turkish highway map. In this demonstration, customers correspond to the cities and we assume that there is a main collection center at Ankara, the capital of Turkey. A truck starting at this center collects the goods from each city towards Ankara. In Turkey there are 81 cities. We used the most industrialized 31 cities in our analysis. We also generated a smaller set with 24 cities to test the performance of the model with respect to increase in the number of nodes. We did not include Istanbul in our computational analysis since the volume generated there is large enough for a dedicated trip to Istanbul.

Our model has 5 types of parameters: travel distances between each city pair, the demand of each city, the number of trucks (vehicle), tare and capacity of a truck. For the travel distances, we used the data proposed by Tan & Kara (2007)) which is available at (www.bilkent.edu.tr\~bkara/hubloc.htm). For the demand of each city we assumed that each individual generates 10^{-5} kg of reusable materials and so scaled the population of each city with that tare. Truck capacity is taken as 100 units whereas tare has been taken as 15% of the capacity.

We varied the number of trucks by starting from the minimum possible number, which is 3 for both cases, up to the largest meaningful number, which is 8 for 31 cities and 6 for 24 cities. As can be seen in Table 1, increasing the number of trucks to values more than those numbers, results in the cost increasing since the model forces to use that many trucks by assigning dedicated trips to certain cities and so the cost increases even though we increase the number of trucks.

In Table 1, we present the objective function values, and the CPU hours provided by CPLEX. We terminate when the CPU time reaches 24 hours.

# of cities	# of trucks	Energy consumption (kg-km)	CPU time
31	4	383835	24 hr (optimality gap %11)
	5	345462	24 hr (optimality gap %1.69)
	6	332188	4 hr
	7	321156	317.65 sec
	8	317522	27.54 sec
	9	320031	29.29 sec
24	4	311947	22.4 min
	5	284125	31.53 sec
	6	279521	10.32 sec
	7	281313	12.27 sec

Table 1. The CPU times and energy consumption values

For 31 cities, the CPU time increases as the number of trucks decreases. For 4 and 5 trucks, we terminated after 24 hours and reported the optimality gap. However, when the number of trucks is 7 or more, the optimum solutions are obtained within seconds. When we decrease the number of cities to 24, the CPU time decreases drastically. Even the 4 truck case is solved within minutes.

Next, we wanted to observe the effect of the objective function on the optimal routes. For each (# of city, #of truck) combination, we generated two instances: one with the energy minimizing objective, and the other with the distance minimizing objective (the one customarily used in the literature). For each instance, we report the energy used and the distance of the corresponding solution in Table 2. We also calculate the percent deviation of each value from the best possible (e.g. [Energy consumption of the distance minimizing routes] / [Energy consumption of the energy minimizing routes])

# of city	# of truck	Minimizing Energy			Minimizing Distance		
		Energy consumption (kg*km)	Distance traveled (km)	Percent from the best distance value	Energy consumption (kg*km)	Distance traveled (km)	Percent from the best energy value
31	4	383835	9056	1,13	450907	8012	1,17
	5	345462	9517	1,15	424602	8304	1,23
	6	332188	10511	1,22	434119	8649	1,30
	7	321156	11379	1,25	421770	9115	1,31
	8	317522	11914	1,23	437255	9683	1,38
24	4	311947	8423	1,16	401120	7284	1,29
	5	284136	8857	1,15	374798	7677	1,32
	6	279521	9656	1,17	343374	8269	1,23

Table 2. The results under both scenarios

As can be seen in Table 2, the energy consumption of the routes which minimizes the total distance traveled can be up to 38% (31 cities, 8 trucks case) over the best possible energy consumption. Meanwhile, minimizing the energy consumption can increase the total distance traveled as much as %25.

Also observe from Table 2 that, when the objective is total distance traveled, the smallest possible and largest meaningful numbers of trucks are equal (4 for both 24 and 31 cities). When we increase the number of trucks to 5 or more, the total distance traveled increases as the solution includes dedicated trips to certain cities (the ones closer to main depot Ankara). However, when the objective also includes the weights of the trucks, the cost could be decreased by the addition of the new trucks. The figures 2 and 3 demonstrate the solutions for 31 cities and 5 trucks.

Observe from the figures 2 and 3 that the routes of the two cases are very different from each other. The eastern cities generate smaller volume and so 10 of them can be served by one truck (Figure 3). However, when we include the weights in the objective, the two routes are split into three routes.

Even though we proposed a model with $O(n^2)$ binary variables and $O(n^2)$ constraints for the CumVRP, the CPU times of CPLEX over moderate sized problems were not promising. It is

therefore necessary to develop efficient solution procedures for each special cases of the CumVRP, like heuristics proposed for CVRP (Gendreau, et al.,(1994); Toth & Vigo(2003)). However, these modifications are beyond the scope of this paper.

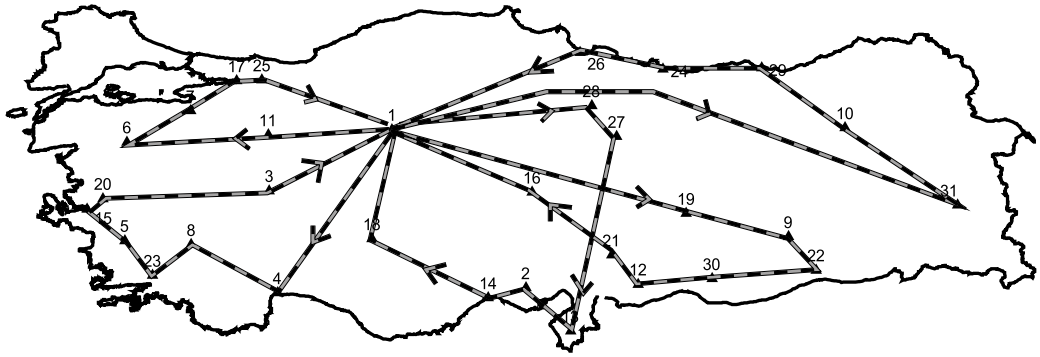


Figure 2. The energy minimizing routes with 5 trucks

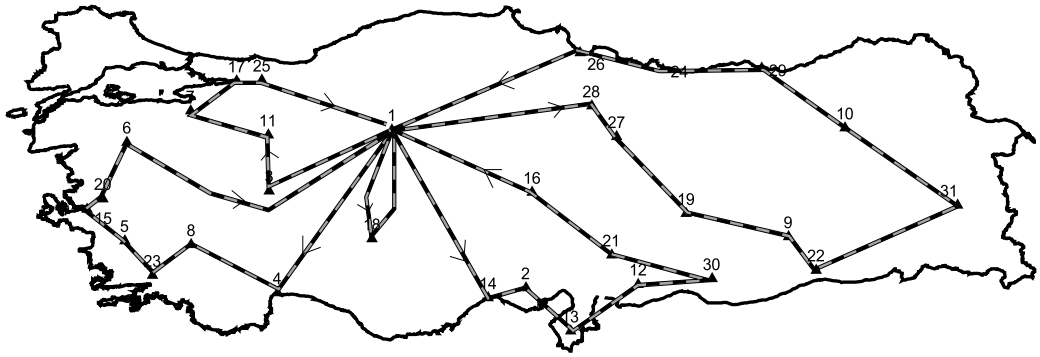


Figure 3. The distance minimizing routes for 5 trucks

5. Conclusion

This paper proposes a new objective function and corresponding formulations for the vehicle routing problem. The new cost function defined as the product of the distance of the arc and the flow on that arc. We call a vehicle routing problem with this new objective function as the Cumulative Vehicle Routing Problem (CumVRP). Integer programming formulations with $O(n^2)$ binary variables and $O(n^2)$ constraints are developed for both collection and delivery cases. We show that the CumVRP is a generalization of the m-Traveling Repairman and related problems in the literature; as an additional finding, we propose an integer programming formulation with $O(n^2)$ constraints and decision variables for the m-Traveling Repairman Problem. We discuss two additional applications of the CumVRP: the Energy-Minimizing VRP and the Average Distance-Minimizing School-Bus Routing Problem.

The collection case of the proposed models for Energy Minimizing case of the CumVRP are tested and demonstrated by using CPLEX 8.1 on some problems from Turkey's 31 and 24

city distance data. We conclude that, increasing the number of vehicles up to a threshold value causes a decrease in the total energy used. We also observed that, CPU time increases as the number of the vehicles decreases. As expected, the number of the customer, i.e., the number of the nodes of the related network, effects CPU times directly. Distance minimizing and energy minimizing solutions of the same problem indicate that, energy minimizing routes travel over longer distances than distance minimizing solutions and optimal routes of the distance minimizing case consume more energy than the other.
Developing good heuristics for each special case of CumVRP and to conduct a computational analysis for comparing integer programming formulations of m-TRP problems are possible future research extensions.

6. Acknowledgment

The authors would like to express their sincere thanks to Gilbert Laporte for his valuable comments and suggestions on an earlier version of this manuscript.

7. References

N.R. Achutan, L. Caccetta, and S.P. Hill. (1996). A new subtour elimination constraint for the vehicle routing problem, *European Journal of Operational Research* 91 573-586.

A. Archer, A. Levin and D. P. Williamson.(2003). Faster approximation algorithms for the minimum latency problem, *in Proceedings of 14th Symposium of Discrete Algorithms(SODA)*, 88-96

R. Baldacci, E. Hadjiconstantinou and A. Mingozzi. (2004) .An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation, *Operations Research* 52 723-738.

L. Bianco, A. Mingozzi and S. Ricciardelli. (1993). The traveling salesman problem with cumulative costs, *Networks*, 23, 81-91.

A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan and M. Sudan. (1994). The minimum latency problem in *STOC*, 163-171.

L.D. Bodin. (1990). Twenty years of routing and scheduling, *Operations Research* 38(4), 571-579.

A. Corberan, E. Fernandez, M. Laguna and R. Marti. (2002). Heuristic solutions to the problem of routing school buses with multiple objectives, *Journal of the Operational Research Society* 53, 427-435.

G.B. Dantzig and J.H. Ramser. (1959). The truck dispatching problem, *Management Science*, 6, 80-91.

M. Desrochers, J. K. Lenstra and M. W. P. Savelsbergh. (1990). A classification scheme for vehicle routing and scheduling problems, *European Journal of Operational Research*, 46, 322-332.

M. Fischetti, G. Laporte and S. Martello. (1993). The delivery man problem and cumulative matroids, *Operations Research*, 41(6), 1055-1064.,

M. Gendreau, A. Hertz, and G. Laporte. (1994). A tabu search heuristic for the vehicle routing problem, *Management Science*, 40, 1276-1290.

L.Gouveia. (1995). A result on projection for the vehicle routing problem, *European Journal of Operational Research*, 856, 10-624.

L.Gouveia and S. VoB. (1995). A classification of formulations for the (time-dependent) traveling salesman problem, *European Journal of Operational Research*, 83, 69-82.

R. Jothi and B.Raghavachari. (2007). Approximating the k-traveling repairman problem with repair times, *Journal of Discrete Algorithms*, 5, 293-303.

İ. Kara, G. Laporte and T. Bektaş. (2004). A note on the lifted Miller-Tucker-Zemlin subtour elimination constraints for the capacitated vehicle routing problem, *European Journal of Operational Research*, 158, 793-795.

I. Kara, B.Y.Kara, and K.Yetis, (2007). Energy minimizing vehicle routing problem. In A.Dress, Y.Xu, and B. Zhu (Eds), Combinatorial Optimization and Applications, LNCS Vol. 4616, pp. 62-71.

G. Laporte. (1992). The vehicle routing problem: An overview of exact and approximate algorithms, *European Journal of Operational Research* , 59 345-358.

G. Laporte and I.H. Osman. (1995) Routing problems: A bibliography, *Annals of Operations Research*, 61, 227-262.

A. N. Letchford and J-J Salazar-Gonzalez. (2006). Projection results for vehicle routing, *Mathematical Programming*, Ser. B 105, 251-274.

L. Li and Z. Fu. The school bus routing problem: a case study, Journal of the Operational Research Society 53(2002) 552-558.

A. Lucena. (1990). Time-dependent traveling salesman problem-The deliveryman case, *Networks*, 20, 753-763.

J.C. Picard and M. Queyranne. (1978). The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling, *Operations Research*, 26(1), 86-110.

T.K. Ralphs, L. Kopman, W.R. Pulleyblank and L. E. Trotter. (2003). On the capacitated vehicle routing problem, *Mathematical Programming Series* B, 94 343-359.

A.J. Swersey and W. Ballard. (1984). Scheduling school buses, *Management Science*, 30(7), 844-853.

P. Tan and B.Y. Kara (2007). "A Hub Covering Model for Cargo Delivery Systems", *Networks*, 49(1), 28-39.

P. Toth and D. Vigo. (2002a). An overview of vehicle routing problems. In: P. Toth and D. Vigo, (editors). The Vehicle Routing Problem. *SIAM Monographs on Discrete Mathematics and Applications*. SIAM, 1-26.

P. Toth and D. Vigo. (2002b) Models, relaxations and exact approaches for the capacitated vehicle routing problem, *Discrete Applied Mathematics* 123, 487-512.

P. Toth and D. Vigo. (2003). The Granular Tabu Search and its application to the vehicle routing problem, *INFORMS Journal on Computing*,15(4), 333-346.

J.N. Tsitsiklis. (1992) Special cases of traveling salesman and repairman problems with time windows, *Networks*, 22, 263-282.

K. M. Walker. (2000). *Applied Mechanics for Engineering Technology* (sixth edition), Prentice Hall.

H. Yaman. (2006). Formulations and valid inequalities for the heterogeneous vehicle routing problem, *Mathematical Programming*, Ser. A ,106 365-390.

7

Enhancing Solution Similarity in Multi-Objective Vehicle Routing Problems with Different Demand Periods

Tadahiko Murata [1,2] and Ryota Itai [3]
[1] *Policy Grid Computing Laboratory,*
[2] *Department of Informatics,*
[3] *Graduate School of Informatics,*
Kansai University
Japan

1. Introduction

In this chapter, we consider vehicle routing problems (VRPs) where the demand of customers varies. We have proposed a problem with two periods of different demand (Murata & Itai, 2005). In each period, we treat the VRPs as multi-objective optimization problems (MOPs). In MOPs, we can handle several objectives such as minimizing total cost for delivery, minimizing maximum cost, minimizing the number of vehicles, minimizing total delay to the date of delivery and so on. Although a set of non-dominated solutions can be searched independently in each period, NDP or HDP, drivers of vehicles prefer to have similar routes in the both periods in order to reduce their fatigue to drive on a different route. We propose a local search that enhances the similarity of routes in NDP and HDP. Simulation results show that the proposed local search can find a similar set of non-dominated solutions in HDP to the one in NDP.

As for the algorithm to find a set of solutions for MOPs, we have various approaches in Evolutionary Multi-criterion Optimization (EMO) community community (Zitzler et al., 2001; Fonseca et al., 2003; Coello Coello et al., 2005; Obayashi et al, 2007). However, there are few research works that investigate the similarity among obtained sets of non-dominated solutions. Deb (2001) considered topologies of several non-dominated solutions in Chapter 9 of his book. He examined the topologies or structures of three-bar and ten-bar truss. He showed that neighboring non-dominated solutions on the obtained front were under the same topology, and NSGA-II could find the gap between the different topologies. While he considered the similarity of solutions in a single set of non-dominated solutions from a topological point of view, there is no research work relating to EMO that considers the similarity of solutions in different sets of non-dominated solutions. In this chapter, we propose a local search in an EMO algorithm that enhances the similarity of solutions in different sets of non-dominated solutions.

2. Multi-objective vehicle routing problems

The VRP is a complex combinatorial optimization problem that can be seen as a merge of two well-known problems: Traveling Salesman Problems (TSPs) and Bin Packing Problems (BRPs). This problem can be described as follows: Given a fleet of vehicles, a common depot, and several customers scattered geographically. Find the sets of routes for the fleet of vehicles. As for objective functions considered in VRPs, many research works (Tavares et al., 2003; Berger & Barkaoui, 2003; Tan et al., 2003; Saadah et al., 2004; Chitty & Hernandez, 2004) on their VRP try to minimize the total route cost that is calculated using the distance or the duration between customers. Tan *et al.* (2003) and Saadah *et al.* (2004) employed the travel distance and the number of vehicles to be minimized. Chitty & Hernandez (2004) tried to minimize the total mean transit time and the total variance in transit time.

In this chapter, we employ three objectives. One is to minimize the maximum routing time and another is to minimize the number of vehicles in VRPs. It should be noted that we don't employ the total routing time of all the vehicles, but use the maximum routing time among the vehicles. We employed it in order to minimize the active duration of the central depot of all vehicles. Even if the total routing time is minimized, the central depot should be opened until the last vehicle comes back to the depot. In order to minimize the active duration of the central depot, the maximum routing time should be minimized.

As for the third objective, we consider the maximization of the similarity of solutions. In this chapter, we suppose two periods with different demands. One period has a normal demand of customers. The other has a higher demand. We refer to the former period and the latter period as Normal Demand Period (NDP) and High Demand Period (HDP), respectively. We define the demand in the HDP as an extended demand of the NDP. For example, we assume that the demand in the HDP is a demand occurring in a high season such as Christmas season. In that season, the depot may have an extra demand in addition to the demand in the normal season. In order to avoid big changes of each route from the depot, a solution (i.e., a set of route) in HDP should be similar to a solution in NDP. This situation requires us to consider the similarity of solutions on different non-dominated solutions in multi-objective VRPs.

In order to find a set of non-dominated solutions in the HDP that is similar to a set of non-dominated solutions in the NDP, we apply a two-fold EMO algorithm (Murata & Itai, 2005) to the problem. In the two-fold EMO algorithm, first we find a set of non-dominated solutions for the NDP by an EMO algorithm. In order to enhance the similarity between sets of non-dominated solutions in NDP and HDP, we showed the effectiveness of utilization of a solution set in NDP for population initialization in HDP. The two-fold EMO algorithm is explained in the next section.

The domain of VRPs has large variety of problems such as capacitated VRP, multiple depot VRP, periodic VRP, split delivery VRP, stochastic VRP, VRP with backhauls, VRP with pick-up and delivering, VRP with satellite facilities, VRP with time windows and so on. These problems have the basic architecture of the VRP except their own constraints. Those constraints are arisen in practical cases. For the detail of the VRP problems, see Lensta & Rinnooy Kan (1981).

A solution of the VRPs is represented by a permutation of N customers, and we split it into M parts as shown in Figure 1. It shows eight customers that are served by three vehicles. The first vehicle denoted v_1 in the figure visits three customers in the order of Customers 1,

2 and 3. Each solution is divided by a closed triangle. Therefore the driving duration for v_1 is calculated by $c_{D,1} + c_{1,2} + c_{2,3} + c_{3,D}$. Figure 2 shows an example of three routes depicted on the map of eight customers and the depot. It should be noted that, we consider only problems with symmetric cost where $c_{1,2} = c_{2,1}$ in this chapter.

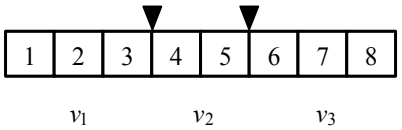


Fig. 1. An example of eight customers visited by three vehicles. Each triangle shows the split between the routes for vehicles.

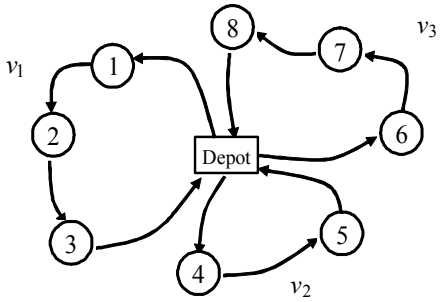


Fig. 2. An example of eight customers visited by three vehicles.

The objective employed in many VRPs is to minimize a total cost described as follows:

$$\text{Min. } \sum_{k=1}^{M} c_k \ , \tag{1}$$

where M is the number of vehicles that start from the depot and are routed by a sequence of customers, then return to the depot. The cost of k-th vehicle is denoted by c_k and described as follows:

$$c_k = c_{D,1} + \sum_{i=1}^{n_k - 1} c_{i,i+1} + c_{n_k,D} \ , \tag{2}$$

where $c_{i,j}$ means the cost between Customers i and j. Let us denote D as the index for the depot in this paper. Equation (2) indicates the sum of the cost between the depot and the first customer assigned to the k-th vehicle (i.e., $c_{D,1}$), the total cost from the 1st customer to the n_k-th customer (i.e., $\sum_{i=1}^{n_k - 1} c_{i,i+1}$), and the cost between the final customer n_k and the depot. Each vehicle is assigned to visit n_k customers, thus we have $N = \sum_{k=1}^{M} n_k$ customers in total. The aim of this VRP is to find a set of sequences of customers that minimizes the total cost. Each customer should be visited exactly once by one vehicle.

While the total cost of all the vehicles is ordinarily employed in the VRP, we employ the maximum cost to be minimized in this paper. When the cost $c_{i,j}$ is related to the driving

duration between Customers *i* and *j* in Equation (2), the total cost c_k for the *k*-th vehicle means the driving duration from the starting time from the depot to the returning time to the depot. In order to minimize the activity duration of the depot, the maximum duration of the vehicles should be minimized since the depot should wait until all the vehicles return to the depot. We also consider the minimization of the number of vehicles in our multi-objective VRP. The objectives in this paper can be described as follows:

$$\text{Min. } \max_k c_k \,, \tag{3}$$

$$\text{Min. } M \,. \tag{4}$$

When we have a solution with $M = 1$, our problem becomes the TSP. In that case, the other objective, to minimize the maximum driving duration in Equation (3), becomes just to minimize the total driving duration by one vehicle. On the other hand, the maximum driving duration becomes minimum when the number of vehicles equals to the number of customers (i.e., $M = N$). In that case, each vehicle visits only one customer. The driving duration for each vehicle in (2) can be described as follows:

$$c_k = c_{D,[1]_k} + c_{[1]_k,D} \,, \tag{5}$$

where $[1]_k$ denotes the index of the customer visited by the *k*-th vehicle. The maximum driving duration in Equation (5) over *M* vehicles becomes the optimal value of that objective in the case of $M = N$. Therefore we face the trade off between these two objectives: the minimization of the maximum driving duration and the minimization of the number of vehicles.

We consider two periods with different demands: NDP and HDP. In NDP, a normal demand of customers should be satisfied. On the other hand, extra demands should also be satisfied in HDP. In this chapter, we increase the number of customers in HDP. That is, $N_{NDP} < N_{HDP}$, where N_{NDP} and N_{HDP} are the number of customers in NDP and HDP, respectively. We can obtain a set of non-dominated solutions for each problem. We refer a set of non-dominated solutions for NDP as Ψ_{NDP}, and that for HDP as Ψ_{HDP}. These two sets of non-dominated solutions can be obtained by applying one of EMO algorithms such as NSGA-II (Deb et al, 2002). But if we apply the algorithm to each of NDP and HDP independently, we can not expect to obtain a set of solutions that is similar to each other.

3. Similarity between sets of non-dominated solutions

In this section, we define a similarity of a non-dominated solution in Ψ_{HDP} obtained for HDP to the set of solutions Ψ_{NDP} for NDP. Since the aim of measuring the similarity is to find a solution in HDP that is similar to one in NDP, we measure the similarity of a solution in HDP to the one in NDP. We measure it by a ratio of the number of the same edges to the number of all edges in a solution of NDP.

We define the similarity of solution x in HDP is as follows:

$$similarity(x) = \max_{y \in \Psi_{NDP}} (similarity(x,y)) = \max_{y \in \Psi_{NDP}} \left(\frac{sames(x,y)}{edges(y)}\right), \; x \in \Psi_{HDP} \,, \tag{6}$$

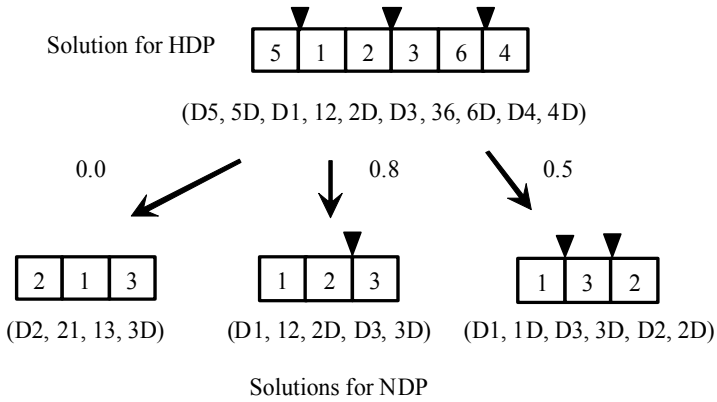


Fig. 3. The similarity of a solution for HDP that is calculated as the maximum similarity among three similarities.

where $similarity(x, y)$ is the similarity of the solution x to the solution y, that is calculated by $sames(x, y)$ (i.e., the number of the same edges) and $edges(y)$ (i.e., the number of edges in a solution y). Figure 3 shows an example to calculate the similarity of solution x (5, 1, 2, 3, 6, 4 with four vehicles) to three non-dominated solutions (2, 1, 3 with one vehicle, 1, 2, 3 with two, and 1, 3, 2 with three) obtained for NDP. The similarity of solutions x becomes the maximum similarity 0.8 through the calculation.

4. Two-fold EMO algorithm for multi-objective VRPs

In this section, we show a Two-Fold EMO algorithm for our multi-objective VRPs (Murata & Itai, 2005). Then we show how we apply a two-fold EMO algorithm to obtain a similar set of solutions in NDP and HDP.

4.1 Genetic operators

[Crossover]

We employ the edge exchange crossover (EXX) (Maekawa et al. 1996) as a crossover operator. This crossover produces offspring only by exchanging edges in parents chromosome, where an edge means a segment between two customers. Therefore offspring chromosomes preserve segments between customers well. The following is the algorithm of this crossover:

Step 1: Select an edge randomly from one parent (Parent 1), and let i_1 be the position of the edge. Let i_2 be the position of the edge of the other parent (Parent 2) whose origin customer is the same as that of the i_1-th edge in Parent l.

Step 2: Let j_2 be the position of the edge of Parent 2 whose origin customer is the same as the destination customer of the i_1-th edge in Parent l, and j_1 be the position of the edge of Parent l whose origin customer is the same as the destination customer of the i_2-th edge in Parent 2.

Step 3: Exchange the i_1-th edge of Parent l and the i_2-th edge of Parent 2. If the destination customers of them are the same, terminate the algorithm.

Step 4: Invert the order of the edges and their origin and destination customers of Parent 1 between the positions i_1 and j_1, and those of Parent 2 between the positions i_2 and j_2.

Step 5: Let $i_1 = j_1$ and $i_2 = j_2$ and go to Step 2.

Figure 4 shows the above procedure between the following parents with one vehicle:
Parent 1: (1 2 3 4 5 6 7 8), and Parent 2: (2 5 4 1 6 7 3 8).
Their edges can be represented as follows:
Parent 1: (12 23 34 45 56 67 78 81), and Parent 2: (25 54 51 16 67 73 38 82).
As an example where the edge 23 of Parent 1 is taken as the starting edge in Step 1 of the above procedure. We have the following offspring after the crossover operation:
Offspring 1: (1 2 5 4 3 8 7 6) , and Offspring 2: (7 3 2 8 1 4 5 6).
In this chapter, we consider any chromosome with multiple vehicles as that with one vehicle. Thus the following cases have the same result in the order of the customers while their positions of Depot do not change between parent and offspring.
Case A: Parent 1: (1 2 | 3 4 5 6 | 7 8), and Parent 2: (2 5 4 1 | 6 7 3 8).
Case B: Parent 1: (1 2 3 | 4 5 6 7 8), and Parent 2: (2 | 5 4 | 1 6 7 | 3 8).
Case A: Offspring 1: (1 2 | 5 4 3 8 | 7 6), and Offspring 2: (7 3 2 8 | 1 4 5 6).
Case B: Offspring 1: (1 2 5 | 4 3 8 7 6), and Offspring 2: (7 | 3 2 | 8 1 4 | 5 6).

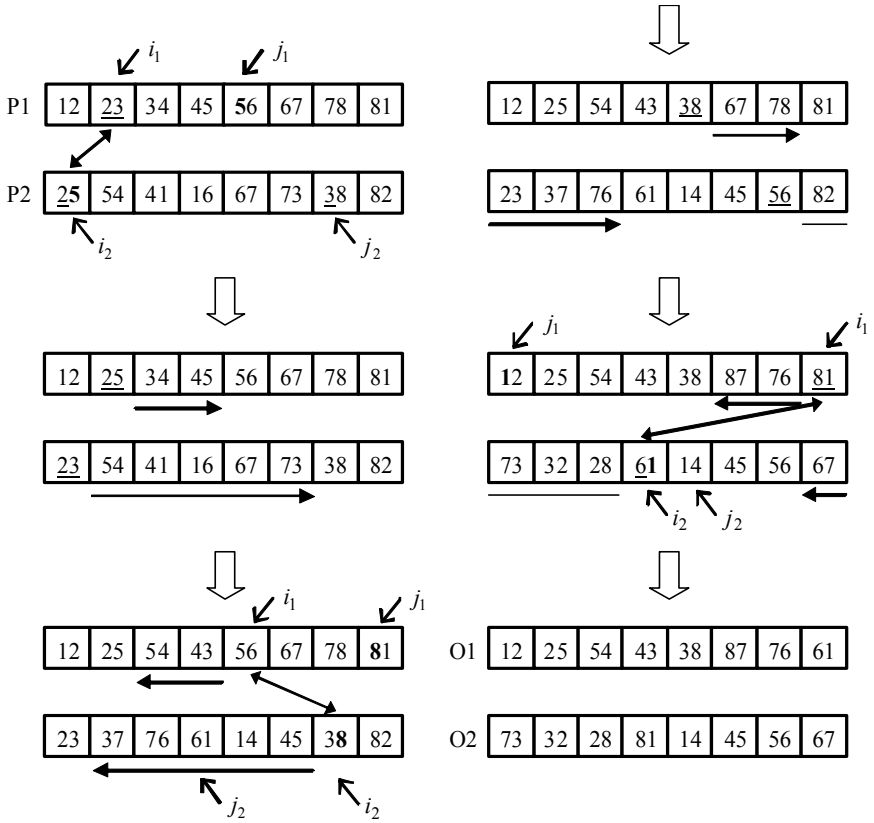


Fig. 4. Examples of Edge Exchange Crossover (Maekawa et al., 1996).

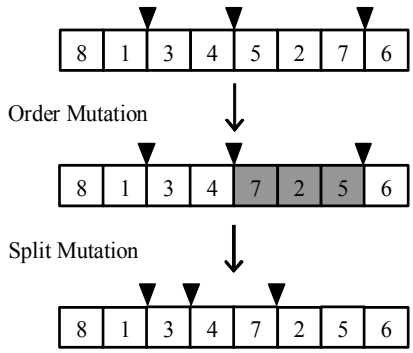


Fig. 5. Examples of two mutation operators. In the order mutation, a selected route is inversed its order of customers. In the split mutation, locations of splits are changed randomly.

[Mutation]
As for the mutation, we employ two kinds of operators in order to modify the order of customers and the locations of splits in a selected route. Figure 5 shows examples of these mutations. It should be noted that the order mutation itself does not affect the two objectives (i.e., the maximum driving duration and the number of vehicles). But it can be useful to increase the variety of solutions when it is used with the crossover and the split mutation.
It should be noted that through crossover and mutation in this chapter, the number of vehicles does not change. Therefore if there is no individual with a certain number of vehicles, no solution with that number of vehicles is generated through genetic search.

4.2 Two-fold EMO algorithm

In our multi-objective VRP, we have two periods, NDP and HDP. Since HDP has extra demands of customers with the demands of NDP, we have two approaches to search a set of non-dominated solutions for each of NDP and HDP. One approach is to apply an EMO algorithm individually to each of them. The other is to apply a two-fold EMO algorithm (Murata & Itai, 2005) to them. In the two-fold EMO algorithm, first we find a set of non-dominated solutions for the NDP by an EMO algorithm. Then we generate a set of initial solutions for the HDP from the non-dominated solutions for the NDP. We apply an EMO algorithm to the HDP with initial solutions that are similar to those of the NDP problem. In our former study (Murata & Itai 2007), we showed that the two-fold EMO algorithm has the better performance than applying an EMO algorithm individually. The procedure of the two-fold EMO algorithm is described as follows:

[Two-Fold EMO Algorithm]

Step 1: Initialize a set of solutions randomly for the NDP. The number of vehicles and the order of customers in each solution are defined randomly.
Step 2: Apply an EMO algorithm to find a set of non-dominated solutions for the NDP until the specified stopping condition is satisfied.
Step 3: Obtain a set of non-dominated solutions for the NDP.
Step 4: Initialize a set of solutions for the HDP using a set of non-dominated solutions of the NDP.
Step 5: Apply an EMO algorithm to find a set of non-dominated solutions for the HDP until the specified stopping condition is satisfied.
Step 6: Obtain a set of non-dominated solutions for the HDP.

In Step 4, we initialize a set of solutions as follows:
Step 4.1: Specify a solution of the set of non-dominated solutions of the NDP.
Step 4.2: Insert new customers randomly into the solution.
Step 4.3: Repeat Steps 4.1 and 4.2 until all solutions in the set of non-dominated solutions of the NDP are modified.

It should be noted that the number of vehicles of each solution is not changed by this initialization. Using this initialization method, we found that the similarity between non-dominated solutions for the NDP and those for the HDP can be increased (Murata & Itai, 2005).

We applied the two-fold EMO algorithm to a VRP that has five customers in NDP and ten customers in HDP. As for an EMO algorithm, we employed NSGA-II (Deb et al., 2002). Figure 6 shows the results of the two-fold EMO, and the EMO applied the HDP with a population initialized randomly. In this problem, we consider only two objectives: the maximum duration and the number of vehicles. We obtained the average maximum duration of a set of non-dominated solutions over 100 trials. We calculate the average similarity after obtaining a set of non-dominated solutions for HDP. In the first figure of Figure 6, we can find that the two-fold EMO can find better non-dominated solutions with

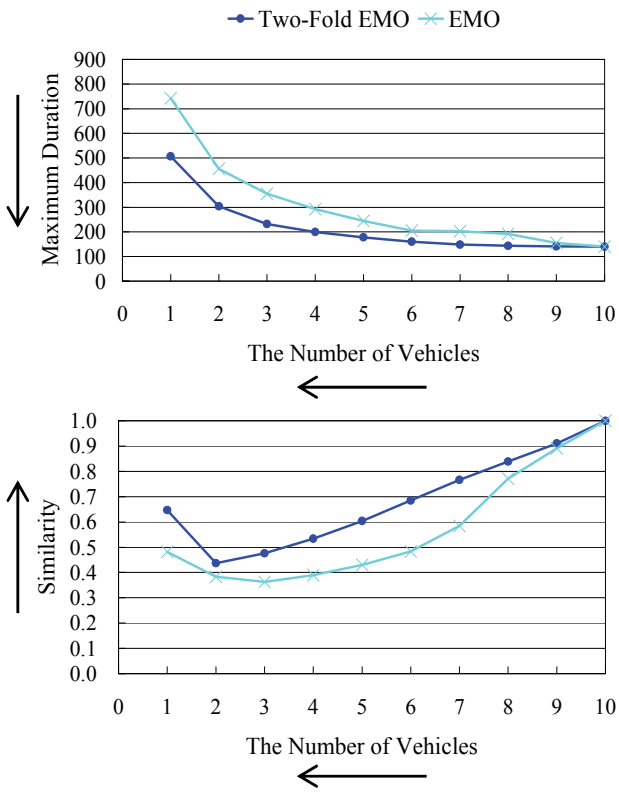


Fig. 6. The obtained non-dominated solutions in the HDP with ten customers. The number of vehicles and the maximum duration are to be minimized, and the similarity to be maximized.

respect to the minimization of the maximum duration and the number of vehicles. From the second figure, we can find that the similarity of non-dominated solutions obtained by the two-fold EMO algorithm is better than that obtained by the EMO algorithm. In this experiment, we can see that improving solutions with respect to the maximum duration does not lead to deterioration of the similarity of non-dominated solutions to those in NDP. Therefore we can say that the two-fold EMO could find the better solutions compared to the EMO for HDP without initial solutions from NDP.

5. Two-fold memetic EMO algorithm

We propose a local search that enhances the similarity of non-dominated solutions for HDP. In order to increase the similarity of a solution for HDP, we incorporate segments between customers from a solution of NDP to a solution of HDP. Therefore we introduce this procedure in an EMO search for HDP not for NDP. The algorithm of the proposed local search can be described as follows:

[Local Search Algorithm]

Step 1: Select a solution x from the current Ψ_{HDP}.

Step 2: Select a non-dominated solution y from Ψ_{NDP} that is used for the calculation of the maximum similarity of x.

Step 3: Select an edge between two customers in y. Note that the edge should be selected within a vehicle.

Step 4: Find a first customer of the selected edge in x.

Step 5: Incorporate the edge to x at the position of the first customer in x. Since the following customer to the first customer in x is replaced by the second customer in the edge, a repairing process should be followed. Find the second customer in x, and replace that with the following customer.

Step 6: Return to Step 3 until all edges in y is incorporated in x.

Figure 7 shows an example of this local search. We apply this local search to each solution of the current set of non-dominated solutions. Since this local search process is introduced to an EMO search in HDP, the two-fold memetic EMO algorithm can be depicted as Figure 8.

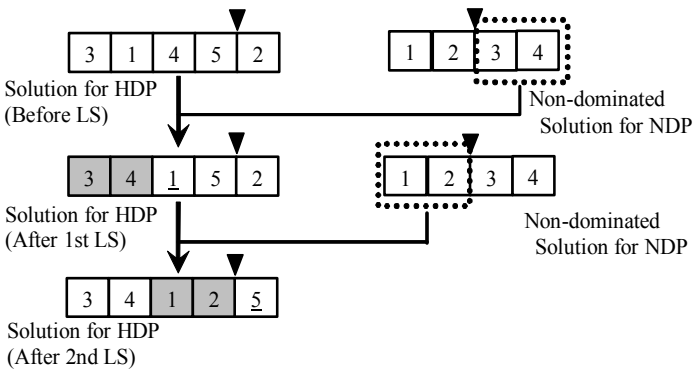


Fig. 7. Local search applied to a solution for HDP. An edge (3, 4) in a non-dominated solution for NDP is incorporated to a solution for HDP. Since (3, 1) in the solution for HDP is replaced with (3, 4), the customers 1 and 4 in the solutions for HDP should be exchanged in a repairing process. Since the solution for NDP has two edges in its string, the local search process terminates at the second time.

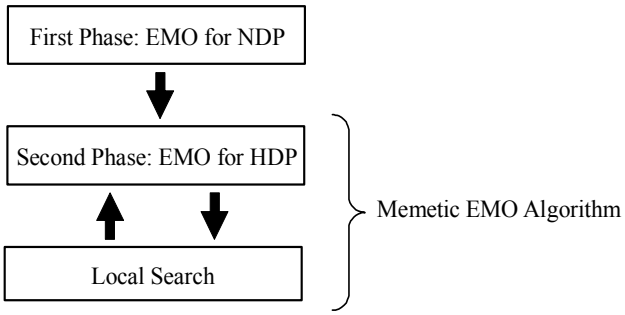


Fig. 8. Two-fold memetic EMO algorithm. A local search is introduced in the second phase of EMO search for HDP.

Number of population	30
Crossover rate	1.0
Order mutation rate	0.04
Split mutation rate	0.02
Terminal generation	2000

Table 1. The parameter specifications in EMO algorithms.

6. Simulation results by two-fold memetic EMO algorithm

We show the simulation result on a multi-objective VRP with NDP and HDP. In that problem, there are five customers in NDP, and ten customers in HDP. Table 1 shows the parameter specifications in our two-fold memetic EMO algorithm. We apply our two-fold memetic EMO algorithm to the problem with 100 different initial solution sets. That is, we obtain average results over 100 trials in a problem. In this section, first we examine the effect of introducing the similarity as third objective. Then we show the effectiveness of the proposed local search to enhance the similarity.

6.1 Effect of similarity

We apply two EMO algorithms to a problem in HDP. One is the two-fold EMO algorithm with three objectives (2F-EMO-3). The other is the two-fold EMO algorithm with two objectives (2F-EMO-2). We don't employ the proposed local search in this section. The result obtained by 2F-EMO-2 is the same that obtained by Two-Fold EMO in Figure 6. We calculate the similarity of non-dominated solutions obtained by 2F-EMO-2 after the search. Figure 9 shows the simulation results obtained by these algorithms. Since the 2F-EMO-3 finds non-dominated solutions on the surface of three objectives, we project them onto the two-objective space in Figure 9. Therefore they are projected between two lines. We depicted two lines of extreme cases, that are the lowest and the highest similarity on the space with the maximum duration and the number of vehicles. On the other hand, the shortest and the longest maximum duration on the space with the similarity and the number of vehicles. From Figure 9, we can see that slightly better solutions are obtained by the 2F-EMO-2 with respect to the minimization of the maximum duration. But it finds worse solutions with respect to the maximization of the similarity. As for the 2F-EMO-3, it

produces slightly better non-dominated solutions in the similarity when their maximum duration becomes near to those of the 2F-EMO-2. On the other hand, when the 2F-EMO-3 sacrifices the minimization of the maximum duration, the similarity of non-dominated solutions becomes much better than the 2F-EMO-2. Through this figure, we can find that the similarity of non-dominated solutions has the trade-off relationship with the maximum duration. Therefore the introduction of the similarity as the third objective is needed for those who wants to have similar routes in HDP to NDP.

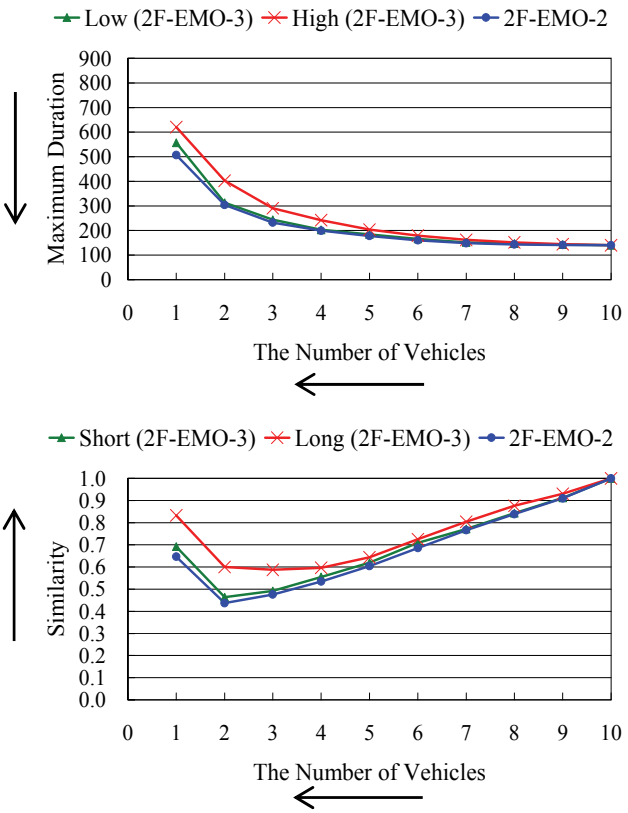


Fig. 9. The effect of the similarity.

6.2 Effect of local search to enhance the similarity in HDP

In this section, we examine the effectiveness of the proposed local search to enhance the similarity of non-dominated solutions for HDP. We compare the 2F-EMO-3 and the two-fold memetic EMO algorithm (2F-mEMO). From Figure 10, We can see that the 2F-EMO-3 could find better solutions with respect to the maximum duration when it sacrifices the similarity. On the other hand, almost similar maximum durations are obtained by both algorithms when they seek to maximize the similarity. Although both the algorithms have similar maximum durations in the case of high similarity, the degree of the similarity of these algorithms is quite different in the latter figure of Figure 10. Using the proposed local search, the 2F-mEMO could enhance the similarity especially in non-dominated solutions

with two through six vehicles. As for the solutions with more than seven vehicles, the similarity is not improved well. This is because each vehicle should not visit several customers when the number of vehicles is similar to the number of customers. Similar routes are required when each vehicle has several customers to visit. From Figure 10, we can see that the proposed local search is very much effective in enhancing the similarity with a slight deterioration in the maximum duration.

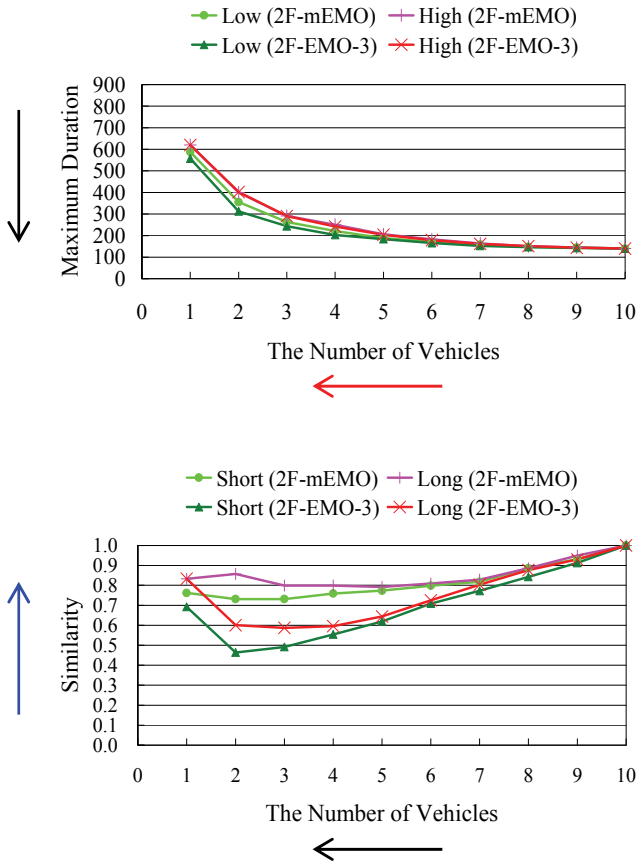


Fig. 10. The effect of the local search in HDP.

7. Conclusion

In this chapter, we proposed a local search that can be used in a two-fold EMO algorithm for multiple-objective VRPs with different demands. The simulation results show that the proposed method have the fine effectiveness to enhance the similarity of obtained routes for vehicles. Although the local search slightly deteriorates the maximum duration, it improves the similarity of the routes that may decrease the possibility of getting lost the way of drivers. If drivers get lost their ways during their delivery, the cost of his routes may increase. The enhancing the similarity of set of non-dominated solutions seems important when we apply EMO algorithms to practical problems.

Since the algorithm of the proposed local search to enhance the similarity depends on the problem specifications, we should make further research on the similarity of a set of non-dominated solutions with different problems. We may define similarity on the genotype, and it on the phenotype. Since the similarity on the phenotype may depend on problems, we should research further on the similarity on the genotype of various problems.

8. Acknowledgments

This work was partially supported by the MEXT, Japan under Collaboration with Local Communities Project for Private Universities starting 2005.

9. References

Berger, J., Barkaoui, M. (2003). A hybrid genetic algorithm for the capacitated vehicle routing problem, *Proc. of Genetic and Evolutionary Computation Conference 2003*, pp. 646-656.

Coello Coello, C. A., Hernandez Aguirre, A., Zitzler, E. (2005). *Proc. of Third International Conference on Evolutionary Multi-Criterion Optimization*.

Chitty, D. M., Hernandez, M. L. (2004). A hybrid ant colony optimisation technique for dynamic vehicle routing, In Proc. of Genetic and Evolutionary Computation Conference 2004, 48-59

Deb, K. (2001) Applications of multi-objective evolutionary algorithms. *Multi-objective Optimization Using Evolutionary Algorithms* (John Wiley & Sons, England), 447-479.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. on Evolutionary Computation*, Vol. 6, No. 2, 182-197.

Fonseca, C. M., Fleming, P. J., Zitzler, E., Deb, K., Thiele, L. (2003). *Proc. of Second International Conference on Evolutionary Multi-Criterion Optimization*.

Lenstra, J. K., Rinnooy Kan, A.H.G. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, Vol. 11, 221-227.

Maekawa, K., Mori, N., Tamaki, H., Kita, H., Nishikawa, H. (1996). A Genetic Solution for the Travelling Salesman Problem by Means of a Thermodynamical Selection Rule. *Proc. of 1996 IEEE International Conference on Evolutionary Computation*, 529-534.

Murata, T., Itai, R. (2005). Multi-objective vehicle routing problems using two-fold EMO algorithms to enhance solution similarity on non-dominated solutions, *Proc. of Third International Conference on Evolutionary Multi-Criterion Optimization,* 885-896.

Murata, T., Itai R. (2007). Local search in two-fold EMO algorithm to enhance solution similarity for multi-objective vehicle routing problems, *Proc. of Fourth International Conference on Evolutionary Multi-Criterion Optimization,* 201-215.

Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (2007). *Proc. of Third International Conference on Evolutionary Multi-Criterion Optimization*.

Saadah, S., Ross, P., Paechter, B. (2004). Improving vehicle routing using a customer waiting time colony, *Proc. of 4th European Conf. on Evolutionary Computation in Combinatorial Optimization*, 188-198.

Tan, K. C., Lee, T. H., Chew, Y. H., Lee, L. H. (2003). A hybrid multiobjective evolutionary algorithms for solving vehicle routing problem with time windows, *Proc. of IEEE International Conf. on System, Man, and Cybernetics 2003*, 361-366.

Tavares, J., Pereira, F. B., Machado, P., Costa, E. (2003). Crossover and diversity: A study about GVR, In *Proc. of AdoRo (Workshop held at Genetic and Evolutionary Computation Conference 2003)*, 7 pages.

Zitzler, E., Deb, K., Thiele, L., Coello Coello, C. A., Corne, D. (2001). *Proc. of First International Conference on Evolutionary Multi-Criterion Optimization*.

8

A Multiobjectivization Approach for Vehicle Routing Problems

Shinya Watanabe[1] and Kazutoshi Sakakibara[2]
[1]*Muroran Institute of Technology,*
[2]*Ritsumeikan Univ.,*
Japan

1. Introduction

In this chapter, we describe a new approach for vehicle routing problems (VRPs), which treats VRPs as multi-objective problems using the concept of multiobjectivization. The multiobjectivization approach has generated in the field of Evolutionary Multi-Criterion Optimization (EMO) (Knowles et al., 2001). This approach transforms a single-objective problem into a multi-objective problem. It is most important feature of this approach to provide more freedom to explore and to reduce the likelihood of becoming trapped in local optima by adding additional objectives.

There have been many studies using EMO algorithm to optimize multi-objective VRPs, with objectives including the number of routes and total travel distance or number of routes and duration of routes, etc (Jozefowiez et al., 2002). In these studies, EMO treats the original objective of VRPs directly as multi-objective.

On the other hand, our approach deals not only with the original objective of VRPs but also with newly defined objectives related to assignment of customers. Generally, VRPs seem to have two different determinations: the assignment of customers and the order of the route. The assignment of customers is known to have a stronger influence on the search than the order of the route in many studies (Doerner et al., 2005). Therefore, we expect that the proposed approach will get better solutions in minimization of the total travel distance than the approach using only the total travel distance as a single objective. In our approach, we add two new objective functions used by MOCK (Handl & Knowles, 2005) as the objective related to assignment of customers. Our multiobjectivization aims to accelerate the search for original objective by adding supplementary objective.

We investigated the characteristics and effectiveness of the proposed approach by comparing the performance of the conventional approach and multiobjectivization approach. In numerical experiments, we used Taillard's test functions as a benchmark problem. In addition, we used NSGA-II (Deb et al., 2002) in implementing our approach. Through numerical examples, we showed that the proposed multiobjectivization approach can obtain the solution with good quality and little variation in VRPs.

2. Vehicle routing problem

This paper deals with the most elementary version of VRPs, the capacitated VRPs (CVRPs), which can be described as follows (Braysy & Gendreau, 2005):

- All vehicles start from the depot and visit the assigned customer points, then return to the depot.
- Here, a route is formed by the sequence of the depot and the customer points visited by a vehicle.
- Therefore the number of vehicles is same as the number of route. Moreover, each customer is visited only once by exactly one vehicle.
- Each customer asks for a weight $w_i (i = 1, \ldots, N)$[1] of goods and a vehicle of capacity W is available to deliver the goods. In this paper, we used the same capacity W for all vehicles.
- A solution of the CVRP is a collection of routes where the total route demand is at most W.

VRPs have a number of objectives, such as minimization of the total travel distance, minimization of the number of routes, minimization of the duration of the routes, etc.
In this paper, we used minimization of the total travel distance (F_{sum}) as the objective of the VRPs. The formula of the objective is as follows:

$$\text{minimize } F_{\text{sum}} = \sum_{m=1}^{M} c^m \tag{1}$$

where M is the total number of routes and c^m indicates m th route distance. The formula for c^m is as follows:

$$c^m = c^m_{0,u_1^m} + \sum_{i=1}^{n_m - 1} c^m_{u_i^m, u_{i+1}^m} + c^m_{u_{n_m}^m, 0} \tag{2}$$

where $c^m_{i,j}$ indicates the distance from customer i to customer j. u_i^m represents the i th customer to be routed in the m th route and "0" is the depot. n_m indicates the total number of customers in the m th route. Here, the total number of customers is $N = \sum_{m=1}^{M} n_m$.
VRPs have a constraint on the vehicle capacity W. In this paper, we used the same capacity W for all vehicles.
The formula of the vehicle capacity W is as follows:

$$W \geq w^m = \sum_{i=1}^{n_m} w_{u_i^m} \qquad (m = 1, \ldots, M) \tag{3}$$

where w^m indicates the amount of customers' weight in the m th route and $w_{u_i^m}$ represents the weight of the goods for the i th customer to be routed in the m th route.
As noted above, in VRPs it is necessary to find a set of sequences of customers that will minimize the total travel distance. In addition, it is necessary to determine the following two points:

---

[1] N is the number of customers.

1. assignment of customers
2. routing (the order of customers)

3. The multiobjectivization of vehicle routing problem

In this section, we describe the purpose of multiobjectivization and the evaluation method related to assignment of customers.

3.1 Multiobjectivization approach

The term of multiobjectivization was born in the field of Evolutionary Multi-Criterion Optimization (EMO) (Knowles et al., 2001). The main concept of this approach is to translate single-objective optimization problems into multi-objective optimization problems and then apply EMO algorithm to the translated problem.

Previous studies of multiobjectivization can be divided roughly into two categories as follows(S. Watanabe & K. Sakakibara, 2005):

- Addition of new objectives to a problem.
- Decomposing a problem into sub-problems.

These multi-objectivizations have a number of effects, such as the reduction of the effect of local optima, making the problem easier, or increasing search paths to the global optimum.

Our multiobjectivization approach for VRPs is based on the addition of new objectives. In our approach, newly defined objective aims to accelerate the speed of the search for original objective. Fig.1 shows the concept of this approach.

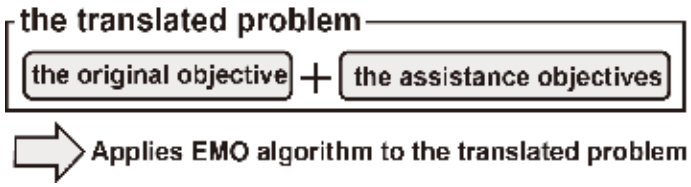


Fig. 1. The concept of multiobjectivization

3.2 The purpose of the proposed multiobjectivization approach

As described in section 2, two types of decision elements should be considered in VRPs. Among two decisions, the assignment of customers has a stronger influence on the search than the order of customers, because the order of customers can be determined under the fixed assignment of customers to the specific vehicle.If the assignment determination is not appropriate, good solutions cannot be obtained even if the best order is determined for all routes.

However, there have been no previous reports of VRPs explicitly taking into account evaluation of customer assignments. As a hierarchical search between the assignments and order determination, Bent et al. proposed a two-stage hybrid local search that first minimizes the number of vehicles using SA, and then minimizes the total travel distance using a large neighbourhood search (Bent & Hentenryck, 2004). In addition, Nanry et al. reported a hierarchical search using Reactive Tabu Search (RTS) to the customer assignments and the order determination (Nanry & Barnes, 2000). However, these approaches do not evaluate the customer assignments directly, and evaluate only the original objective of VRPs: the total travel distance and the number of vehicles.

The proposed approach treats two types of decision elements independently. But this approach uses the same solution strategy for these decisions, not using individual solution strategy as a hierarchical search. Treating VRP as multi-objective problem, we can handle two different decisions concurrently and independently.

3.3 The evaluation method related to assignment of customers

In the proposed approach, we capitalize on the objective functions using ''Multi-objective clustering with automatic determination of the number of clusters (MOCK) (Handl & Knowles, 2005)'' to evaluate customer assignments.

Clusters and data points in clustering problems can be assumed as routes and customers in VRPs, respectively. Therefore, the objective functions used by MOCK can be diverted to evaluation of the customer assignments in VRPs.

MOCK adopts the following two functions, which reflect two fundamentally different aspects of good clustering solutions.

1. The global concept of the compactness of clusters.
2. The local concept of the connectedness of data points.

The first of these clustering objectives evaluates the overall density of clusters as the compactness, and the latter evaluates the degree to which neighbouring data points have been placed in the same cluster as the connectedness.

The overall density of clustering solutions, which reflects the overall intra-cluster spread of the data, is computed as:

$$\mathrm{Dev}(C) = \sum_{C_k \in C} \sum_{i \in C_k} \delta(i, \mu_k) \tag{4}$$

where C is the set of all clusters, μ_k is the centroid of cluster C_k and $\delta(.,.)$ represents the distance function (Euclidean distance).

The second objective function, connectivity, evaluates the degree to which neighbourhood data points have been placed in the same cluster.

The second objective function is presented as the following formula:

$$\mathrm{Conn}(C) = \sum_{i=1}^{N} \left(\sum_{j=1}^{L} x_{i,nn_i(j)} \right), \qquad x_{i,nn_i(j)} = \begin{cases} 1/j & \text{if } \nexists C_k : i, nn_i(j) \in C_k \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $nn_i(j)$ is the j th nearest neighbour of datum i, and L is a parameter determining the number of neighbours that contribute to the connectivity measure. $x_{i,nn_i(j)}$ represents the penalty value related to whether data i and j th nearest neighbour of data i are placed in the same cluster or not.

In Eq.(5), if data i and j th nearest neighbour of data i are not placed in the same cluster , $1/j$ is added as the penalty value $x_{i,nn_i(j)}$. Therefore, greater values of Eq.(5) indicate the tendency of a clustering solution in which neighbouring data are not placed in the same cluster.

In these different objectives, it is very important that the value of $\mathrm{Dev}(C)$ is decreased with increasing number of clusters, while $\mathrm{Conn}(C)$ is increased by increasing the number of

clusters[2]. Therefore, $\mathrm{Dev}(C)$ and $\mathrm{Conn}(C)$ are in a trade-off relationship with the number of clusters.
Here, we examine the effectiveness of multiobjectivization to VRPs in which one or both of the above-mentioned objectives are added to the original objective.

4. Implementation of GA

In this section, we describe the implementation of GA to multi-objective VRPs based on multiobjectivization as described above.

4.1 Gene expression (String representation)

Various coding methods for VRPs have been proposed. In this numerical experiment, we used all routes directly as the genotype. In other words, the genotype in this numerical experiment is the same as the phenotype that represents the order of all routes.
Therefore, there is no need for translation between genotype and phenotype.

4.2 Population initialization

The average number of customers in one route can be calculated using the vehicle capacity W and a weight $w_i \quad (i = 1, \ldots, N)$ for each customer.

In this numerical experiment, the initial population was generated by random sampling that all routes for each individual must be equal to the average number of customers. The initialization process starts by inserting customers one by one into an empty route in random order until the number of customers in the route is equal to the average number of customers.
If an initial solution is not feasible, we used the repair method (stated in Section 4.6) to make it feasible. Therefore, all solutions of the initial population are feasible.

4.3 Crossover

As genotype has the same coding as phenotype, general crossover operators for VRPs, such as PMX, OX and EX, could not be adopted in this numerical experiment. Therefore, we implemented a new crossover operator, Partial Route Inheritance Crossover (PRIC), which aims to inherit as much as possible of the parents' route information.
In PRIC, the first child inherits half the routes of one parent directly, and then the remaining customers that are not in one parent are inherited using the route information of the other parent. The ratio of direct inheritance from parents to children should have strong effects on the search performance. Here, we designed the method to copy the half of routes in one parent directly.
Fig.2 shows the procedure of PRIC, the details of that are described as follows:
Step 1: Selecting two parents (Parent1 and Parent2) randomly.
Step 2: Copying the half of routes in Parent1 to child.

[2] Because it becomes difficult to place near-neighbour data in the same cluster.

Step 3: The remaining customers that are not in Parent1 are inherited by the routes of Parent2. If the number of routes in the child is complete, the simulation goes to Step 4. If not, the simulation is terminated.

Step 4: Until the number of routes in the child becomes equal to the number of routes in Parent1, routes copied in Step 3 are integrated using the following procedure:

Step 4-1: The routes copied in Step 3 are sorted according to increasing number of customers included.

Step 4-2: Each of the routes according to the sorted order is integrated into the nearest route. The distance between routes is calculated using Euclidean distance between the centred coordinates of the route (including the depot). Therefore, the integration of routes is performed between the closest routes related to the centred coordinates. In this integration, customers are added to the bottom of the nearest route.

In Step 4 above, the routes with a small number of customers are integrated to decrease the total number of routes. Since the remaining customers that are not selected in Parent1 are picked out from Parent 2 routes, a lot of routes with small numbers of customers are produced (Step 3 in Fig.1).

PRIC includes not only the effect of the inheritance of parents' routes but also the effects of the integration of routes and the re-shuffling of customers between routes.

4.4 Mutation

In this paper, we used six kinds of operators as mutation:

1. 2-opt*(asterisk) (Braysy & Gendreau, 2005)
2. or-opt (Braysy & Gendreau, 2005)
3. Relocate Operator (Braysy & Gendreau, 2005)
4. Exchange Operator (Braysy & Gendreau, 2005)
5. Integration of different routes into one route
6. Division of a route into two routes

2-opt*(asterisk) swaps sub-routes between different two routes, and or-opt replaces the sub-route with L customers in a random chosen route. Also, Relocate Operator simply moves a customer from one route to another and Exchange Operator swaps two customers in different routes.

We randomly selected one out of the 6 operators as mutation operator at each generation.

4.5 The decision of start and end point in a route

In this paper, the start and end customers in a route are decided in the evaluation phase.

As the decision of start and end customers determines where to insert the depot in the sequence of customers, we used saving method (Braysy & Gendreau, 2005) to insert the depot with the minimum total travel distance. Therefore, the optimal insertion point of the depot in the sequence of customers can be decided.

4.6 Treatment of a solution with constraint violation

As VRPs have the constraint of the vehicle capacity, we should implement a repair method as a constraint handling technique. We used the repair method to divide an infeasible route into two routes. In this technique, a set of customer sequences satisfying the capacity constraint forms one route, and then the remaining customer sequences forms another route. Therefore, all solutions in this example are feasible.

5. Numerical examples

In this study, we investigated the characteristics and effectiveness of the proposed approach by comparing the performance of both the conventional approaches and multiobjectivization approaches.

To verify the effectiveness of multiobjectivization of VRPs, VRP instances provided by Taillard et al.[3] were used. In implementing our proposed approach, we used NSGA-II proposed by Deb et al. (Deb et al., 2002). Table 1 shows the GA parameters.

population size	200
crossover rate	1.0
mutation rate	1/bit length
number of trials	30

Table 1. GA Parameters

5.1 VRPs instances

We used six types of instances: tai75a, tai100d and newly defined four changing the variation coefficient (C) of the customer weight $w_i \quad (i = 1,\ldots,N)$ in tai75a and tai100d[4].

The characteristics of the instance are described in Table 2. Table 2 represents the number of customers(N), the vehicle capacity(W), the average of the customer weight($\overline{w}$), standard deviation($\sigma(w)$) of the customer weight $w_i \quad (i = 1,\ldots,N)$, and the variation coefficient ($C(w)$) of the customer weight.

tai75c(C=0)	75	1122	127	0.0
tai75c(C=0.6)	75	1122	163.2	0.6
tai75c(original)	75	1122	126.9	1.6
tai100d(C=0)	100	1297	136	0.0
tai100d(C=0.8)	100	1297	135.7	0.8
tai100d(original)	100	1297	135.7	1.6
tai75c(C=0)	75	1122	127	0.0

Table 2. Problem Instance

In Table 2, tai75c($C = 0$) and tai75c($C = 0.6$) are the problems of changing the amount of the customer weight $w_i \quad (i = 1,\ldots,N)$ in tai75c(original) so that the amounts of the variation coefficient of customer weights are about 0 and 0.6 respectively (the customer location and the vehicle capacity of these instances are the same as original instance). In the same way, tai100d($C = 0$) and tai100d($C = 0.8$) are modified so that the amounts of the variation coefficient of tai100d(original) are about 0 and 0.8 respectively.

The small value of the variation coefficient indicates that customer weights are homogenized. More homogenized customer weights make it easier to decide the assignment of customers, because simple heuristic approach, in which neighboring

[3] These test problems are available at http://neo.lcc.uma.es/radi-aeb/WebVRP/.

[4] The variation coefficient (C) is the value that standard deviation was divided by the mean value. C is the index that represents the degree of variation.

customers merge into the same cluster, can be worked effectively in this case. In contradiction to this, the higher value of the variation coefficient make it more difficult to assign customer. In this case, the performance related to assignment of customers seems to influence the quality of the obtained solutions more strongly.

5.2 Results and analysis

In this study, we used five types of NSGA-II experiment based on the implementation of objectives (f_1 and f_2). Table 3 shows the 5 experiments. In Table 3, the first objective, which is common to all experiments, is the total travel distance (Eq.(1)).

In this experiment, termination conditions of the instances with 75 customers (tai75c($C = 0$), tai75c($C = 0.6$) and tai75c(original)) was 5000 generations, and those of the instances with 100 customers (tai100d($C = 0$), tai100d($C = 0.8$) and tai100d(original)) was set to 7500 generations.

We performed 30 trials and all results are shown as averages of 30 trials. The results of the 6 instances are shown in Fig. 3. Fig. 3 shows the minimum, maximum and average values in which the objective value represents the total travel distance. Also, Fig. 4 shows the standard deviation of the solutions so as to evaluate the degree of variation of the solutions. In multiobjectivization approaches, the solution with the minimum total travel distance is treated as the final best solution in each trial. In concrete terms, the solutions with the minimum f_1 value of each experiment are used as the final results.

As shown in Fig. 3, the solutions of multiobjectivization (three proposed approaches) were better than those of the conventional approaches. Especially, the difference in quality of the obtained solutions between the conventional and the proposed approaches increased with larger customer size and variation coefficient (C) value.

method	f_1	f_2	f_3	method
Conventional 1	$f^{Eq.1}$	$f^{Eq.1}$	---	Conventional 1
Conventional 2	$f^{Eq.1}$	The number of routes	---	Conventional 2
Proposed 1	$f^{Eq.1}$	$f^{Eq.4}$	---	Proposed 1
Proposed 2	$f^{Eq.1}$	$f^{Eq.5}$	---	Proposed 2
Proposed 3	$f^{Eq.1}$	$f^{Eq.4}$	$f^{Eq.5}$	Proposed 3

Table 3. The four type experiments of NSGA-II

From the width between minimum and maximum in Fig. 3 and the degree of variation in Fig. 4, it was also clear that the proposed approaches can obtain the solution with good quality and little variation at each trial. These results confirmed that proposed multiobjectivization approaches are more effective for VRPs than both the conventional approaches.

The increase in the number of objectives usually degrades the convergence of solutions to the Pareto front. But the multiobjectivization in this paper doesn't show this tendency, since additional objectives don't yield the trade-off relationship between original and additional objectives. In this paper, additional objectives can help to accelerate the search for original objective. The results of Fig. 3 enhance the legitimacy of this inference.

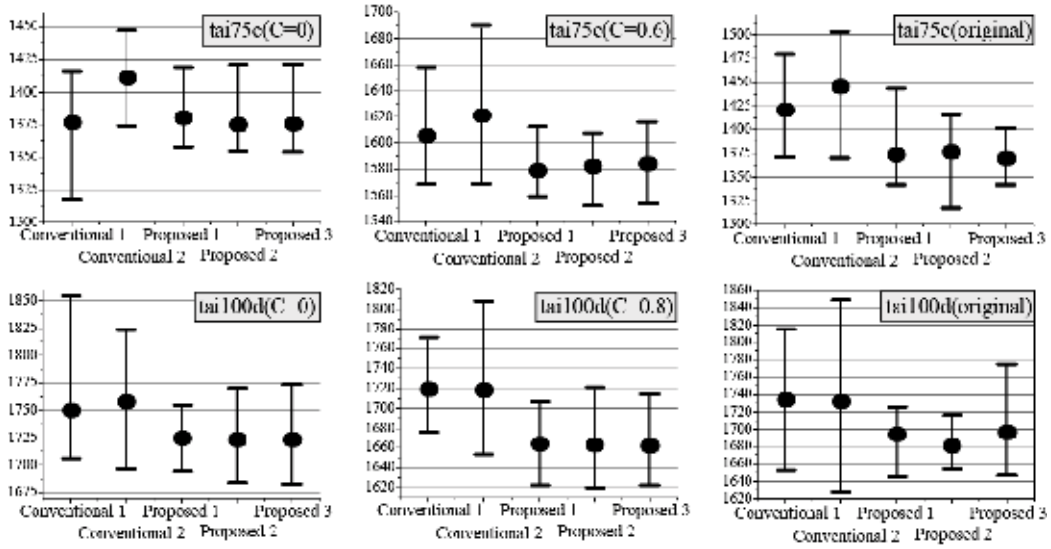


Fig. 3. The results of the total travel distance

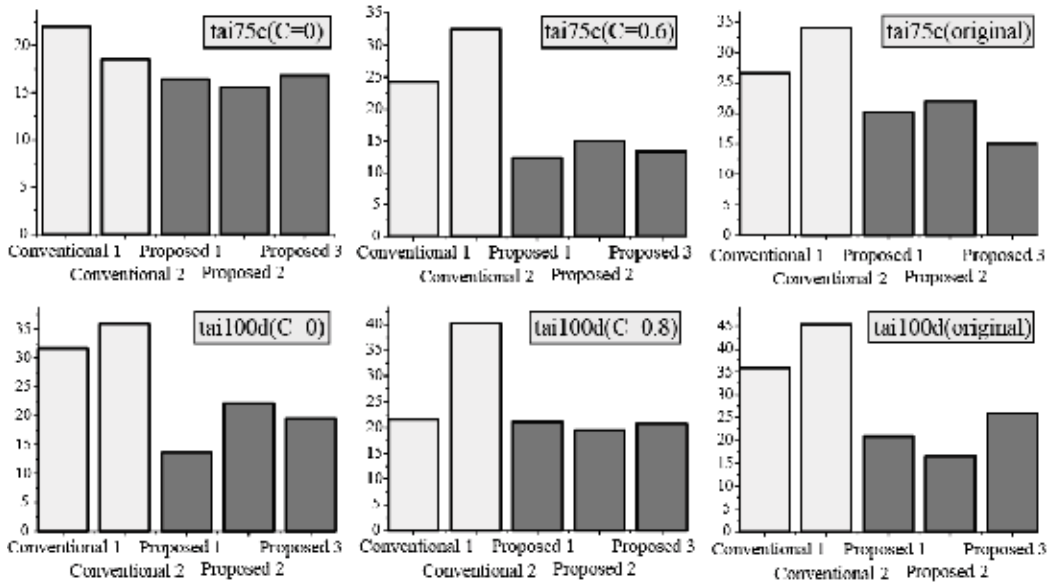


Fig. 4. The standard deviation of the solutions

The transition of the objective values

Here, we describe the transition of the objective values in each approach. The transitions of the objective values in tai100d(original) are shown in Fig. 5. The four objective values in Fig. 5 are the total travel distance(Eq.(1)), the number of routes, compactness ($\mathrm{Dev}(C)$) and connectivity ($\mathrm{Conn}(C)$)). As these objective values of Fig. 5, we used the objective value of

the individuals with minimum value of the total travel distance in the generation[5]. In these figures, the horizontal axes indicate generation and the vertical axes indicate each objective value and the values of generation are described on a log scale (common logarithm). Also, all results are shown as averages of 30 trials.

From Fig. 5, there seem to be some sort of relationships between each objective values, because all objective values were decreased by a large generation number. But the values of compactness and connectivity in three proposed approaches were increased with more than 200 generations. Therefore the correlations of compactness and connectivity with near the minimum value of the total travel distance are guessed to be low or slight trade-off relation.

The transitions of the total travel distance had similar tendencies in all approaches. But the transition of Conventional2 with the total travel distance and the number of routes as objectives were a more slower slope for less than 150 generations as compared to the other approaches. On the other hand, when considering the transition of the number of route, Conventional2 got the smaller value in earlier generations as compared to the other approaches. This was because Conventional2 explicitly evaluate the number of route.

And in terms of the objective value of compactness and connectivity, three proposed approaches yield better results. It is interesting that Proposed3 with both objective functions of MOCK was better than Proposed1 and Proposed2 with one or other of two MOCK functions. Therefore it is apparent from Fig. 5 that Proposed3 is better in terms of assignment of customers than other multiobjectivization approaches with one or other of MOCK functions.

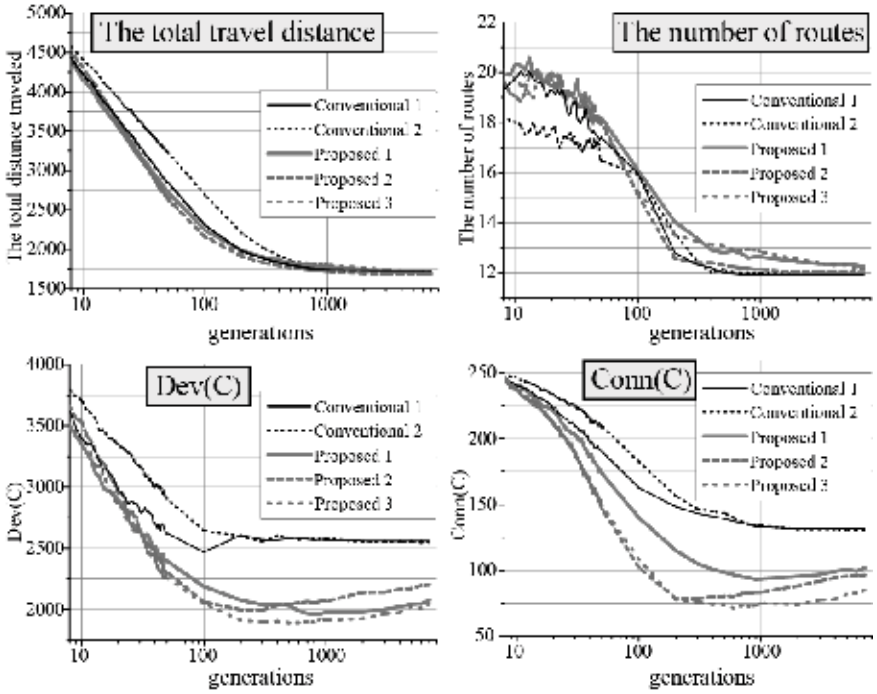


Fig. 5. The transition of the objective values

[5] Therefore, all objective values except the total travel distance may make a change for the worse.

On the other hand, the results concerning compactness and connectivity of Conventional2 with the minimization of the number of routes as second objective function and that of Conventional1 are not so different and worse than that of Proposed3. Therefore, the minimization of the number of routes doesn't have a strong effect on the assignment of customers.

6. Conclusions

In this paper, we proposed a new approach based on multiobjectivization for vehicle routing problems (VRPs) with a single objective. This approach treats VRPs as multi-objective problems in which a newly defined objective related to assignment of customers is added. As the objective related to assignment of customers, we used two objective functions used by MOCK, i.e., the compactness of clusters and the connectedness of data points.
This multiobjectivization aims to accelerate the search for original objective by adding supplementary objective. This approach assumes that the adding objective is empirically-predicted to be useful in improving the objective value. We think that our approach is one of effective means for using the empirical knowledge of the problem.
We investigated the effectiveness of the proposed multiobjectivization approaches by comparison of its performance with that of the conventional approaches.
Numerical experiments clarified the following points:

- Multiobjectivization approaches can obtain the solution with better quality and less variation at each trial. Also, the experimental results indicate that multiobjectivization using both additional objectives is more effective than using either alone.
- From the results of the transition of the objective values in the course of the search process, it was confirmed that the multiobjectivization approach using MOCK functions is very effective for the assignment of customers, while the minimization of the number of routes have little effect on the assignment of customers. Also, the three multiobjectivization approach using both objective functions of MOCK can derive better solutions than other approaches with only one MOCK function.

7. References

R. Bent and P. Van Hentenryck. A two stage hybrid local search for the vehicle routing problem with time windows. In Transportation Science, volume 38, pages 515–530, 2004.

O. Braysy and M. Gendreau. Vehicle routing problem with time windows, part i:Route construction and local search algorithms. Transportation Science, 39(1):104–118, 2005.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation,6(2):182–197, April 2002.

J.Handl and J.Knowles. Exploiting the Trade-Off—The Benefits of Multiple Objectives in Data Clustering. In Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005, pages 547–560, 2005.

K.Deb. Multi-Objective Optimization using Evolutionary Algorithms. Chichester, UK:Wiley, 2001.

K.F.Doerner, R.F. Hartl, and M. Lucka. A parallel version of the d-ant algorithm for the vehicle routing problem. In Theory an Applications, pages 109–118, 2005.

D. Knowles, A. Watson, and W. Corne. Reducing local optima in single-objective problems by multi-objectivization. In Evolutionary Multi-Criterion Optimization. First International Conference, EMO 2001, pages 268–282, 2001.

W. P. Nanry and J. W. Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. Transportation Research Part B, 34:107–121,2000.

N.Jozefowiez, F.Semet, and E.Talbi. Parallel and Hybrid Models for Multiobjective Optimization: Application to the Vehicle Routing Problem. In Parallel Problem Solving from Nature—PPSN VII, pages 271–280, 2002.

J.Y. Potvin and S. Bengio. The vehicle routing problem with time windows - partII: Genetic search. In INFORMS Journal on Computing, volume 8, pages 165–172,1996.

S. Watanabe and K. Sakakibara. Multi-objective approaches in a single-objective optimization environment, 2005 IEEE Congress on Evolutionary Computation (CEC2005),pp.1714-1721, 2005

9

Resources Requirement and Routing in Courier Service[1]

C.K.Y. Lin
Department of Management Sciences, City University of Hong Kong
Hong Kong, P.R.O.C.

1. Introduction

Providing cost-effective and efficient services are important to both courier companies and their contract customers. This work proposes modelling of multi-resource domestic courier operations with the aim of improving the service in terms of meeting a specific time window for pickup/delivery at minimum total cost. The characteristics of this pickup and delivery operations are (a) one delivery resource (e.g., van) can transport both customer items and a non-identical, lighter resource (e.g., courier); (b) item transfer is allowed between resource units where the transfer location(s) is to be decided, say among customer sites. In actual practice of some courier companies, the driver of a van can service customers like couriers. In other occasions, the van is used to transport couriers or return the collected documents in busy districts. Resources estimation and computerized scheduling methods will facilitate contract preparation with customers. Scheduling results can enable regular performance monitoring of daily operations.

This work is motivated by a local courier service of a multi-national logistics company operating with multiple delivery resources (vans, scooters and couriers on foot). Two non-identical resources (vans and couriers) and their two types of cooperation described in (a) and (b) above are modelled and a solution method presented. (In a similar modelling approach, the third resource could also be incorporated, but will not be included here.) In each service session, a given set of N customers is to be served. Each customer is associated with two time window requirements: one for document pickup at the customer site and another for delivery to the mail centre to meet the designated flight departure time. The current problem belongs to the class of static-deterministic pickup and delivery problem with time windows (PDPTW) where there are many pickup points and only a single, common delivery point at the mail centre (referred to as depot here). Pickups occur before delivery to the mail centre. Express documents are usually letters/small parcels, which are not constrained by the transportation capacity of carriers. The objective is to find the minimum cost solution for the two resources to servicing all customers and satisfying the time window constraints in this special case of PDPTW – the many pickup-to-one delivery problem with time window constraints and without capacity constraints. The output includes the number of units required by resource and their routes servicing the customers.

[1] The work described in this paper was fully supported by the Strategic Research Grant from City University of Hong Kong (Project No. 7002261)

Pickup and delivery problems (PDPs) with cooperative operations and transfer opportunity have small coverage in literature, not to mention time windows and multiple non-identical delivery resources. The contribution of this work includes modelling and presenting an exact method for solving a PDPTW with two non-identical resources and cooperative operations described in (a) and (b) above. Comparison with the independent operations will be made on instances generated from real data and simulated data. This work will serve as foundation for developing other exact and heuristic methods in the static or dynamic problem.

2. Literature review

There is a vast amount of literature on pickup and delivery problems and several surveys (e.g., Savelsbergh & Sol, 1995; Ropke & Pisinger, 2006). There is relatively few PDPs on cooperative operations or with transfer opportunity. One of them is a multi-criteria PDP (Shang & Cuff, 1996), allowing transfer of hospital documents between vehicles provided no additional travel time is incurred. A look-ahead heuristic was developed for the dynamic situation as it was reported that there were no exact algorithms to solve similarly sized problems. Transfer operations occurred in express mail services in which vehicles could operate as a feeder-backbone system or/and "am-pm" hub system (Kamoun & Hall, 1996) for large metropolitan area. Each feeder vehicle (small van) assigned to serve a district/route would visit a drop-box periodically to put in the collected documents. A backbone vehicle would move the outbound mail from the drop-box to the sorting facility. In return, the backbone vehicle would collect the inbound mail to be placed in the appropriate drop-box. From there, the feeder starts to deliver the document to its destination. Location of drop-boxes are static and may not be optimal for a given set of demand data. The adoption of transshipment points (like drop-boxes) and the benefit (reduction of total travel distances) were examined in a PDPTW solved by applying a construction and improvement heuristic (Mitrovic-Minic & Laporte, 2006). The locations of transshipment point are tested systematically at several static, convenient locations.
An example of cooperative operations in vehicle routing problems is the truck and trailer routing problem (TTRP), where customers are served by one (or more) of the following three routes: (i) a pure truck route, (ii) a complete vehicle route containing a truck and a trailer as one unit or (iii) a complete vehicle route where the trailer is parked somewhere such that the truck can visit customer locations that are less easily accessible by a complete vehicle. This problem can be formulated as an integer programming model and solved heuristically by a two-phase procedure (Semet, 1995). The first phase assigns trailers to trucks and determines customers to be served by each truck or complete vehicle (truck plus trailer), followed by the second phase of routes generation. Other heuristics for this problem include construction and improvement heuristics (Gerdessen, 1996) ; construction heuristics further improved by specially designed tabu search (Chao, 2002; Scheuerer, 2006) and simulated annealing (Yu et al., 2008). Simplifying assumptions in modelling this complex problem were made, like assuming each trailer is parked exactly once (Gerdessen, 1996); or each customer site can be a candidate parking place (Gerdessen, 1996; Chao, 2002). The assumption of making each customer site a candidate location for documents transfer is also adopted in the present study. The differences are that pickup and delivery time windows are considered here; vehicle capacity constraint can be ignored for express documents. Furthermore, the frequency of cooperation among non-identical delivery resources (for

documents transfer or returning to depot together) is unrestricted, but decided by a model. This work is an extension of solving an uncapacitated PDPTW allowing transfer of documents between units of a single vehicle type (Lin, 2008). Computational results indicate savings in total cost and vehicles over a construction heuristic for a capacitated PDPTW (Lu and Dessouky, 2006) as problem size grows. Besides, such a cooperative strategy (allowing document transfer) with multiple use of vehicles achieve cost savings over the independent strategy. In this work, two operational modes, one independent and one cooperative, of the two resources are analyzed to examine the objective (total cost) value and computational time involved: independent operations; cooperative operations allowing document transfer and return transport (of lighter resource unit(s) by the heavier resource).

The remaining of this chapter is organized as follows. Section 3 describes the model assumptions based on some current practice. Formulation of the independent operations and cooperative operations are given in Section 4 and 5, respectively. Computational experiments based on instances generated from real data and simulated data are presented in Section 6. The last section summarizes the contribution of this work and points to future research areas.

3. Model assumptions

After understanding the practice of a courier service, major operating parameters are collected and model assumptions are made. The majority of customer requests occur on Mondays to Fridays. A day's work is typically divided into two half-day service sessions: morning and afternoon. Customer pickup and delivery time windows are placed in the same session. Hence, each session represents an independent problem. Problem size reduction can also be achieved through clustering locations into independent sub-problems. In their planning, vehicle capacity constraint (on carrying express documents) can be ignored. This assumption was also adopted for the pickup and delivery operations of parcels or medical records (Langevin & Soumis, 1989; Mitrović-Minić et al., 2004; Mitrović-Minić & Laporte, 2004; Shang & Cuff, 1996). Here, a capacity constraint is associated with the heavier resource in carrying the lighter resource unit(s).

For a set of N customers given in a service session, the pickup time window for customer i (i=1,…, N) with pickup time specified at t_i is $[t_i - \delta_P , t_i]$, where a given early allowance δ_P (> 0) is common to all customers and same for each resource. An amount of on-site service time φ is expected at each customer site. The delivery location for all customers is the mail centre at which parcels are processed before outbound delivery. For customer i, the time window for delivery time specified at τ_i is $[0, \tau_i + \delta_D]$, where a given lateness allowance δ_D (> 0) is common to all customers. The assumptions in this work extend from those in an earlier work (Lin, 2008) for a single resource. Additional assumptions characterize the cooperative mode between the two resources. The heavier resource, once assigned a customer group (or route), is assumed to be dominant and the lighter resource will subordinate to its operations. The route and scheduled times of the heavier resource is not delayed by carrying units of the lighter resource. All assumptions are listed as follows:

i. The service session is of duration T.
ii. Each unit of the heavier resource (labelled as resource 2) consists of an operator (also a unit of the lighter resource, labelled as resource 1) and a vehicle. Like resource 1, this composite unit can service customers, transport other units of resource 1 or its collected documents.

iii. The capacity of each resource is not constrained in carrying customer documents.
iv. Resource 2 is constrained in carrying resource 1 units. Each unit of resource 2 can carry C units of resource 1, including the operator of resource 2.
v. The pickup request at each customer location occurs before the delivery request to the common delivery location (depot or mail centre). Both must be serviced by the same resource unit, unless an item is transferred to another resource after pickup. (It is naturally possible to transfer documents between different units of the same resource. This has been modelled in an earlier work (Lin, 2008) and will not be considered here.)
vi. Travel time between a pair of locations (different between the two resources) could be asymmetrical or symmetrical.
vii. The travelling speed of resource k (k = 1, 2) is assumed to be an average of V_k km/hour. (The data on travelling speed will be used to convert the travel distance, collected or simulated, into estimated travel time.)
viii. The unit travelling costs of the two resources are assumed to be proportional to their travelling speed.
ix. Pickup at the customer site should be no earlier than δ_P minutes before the specified pickup time.
x. Waiting is allowed if a resource unit arrives before the earliest pickup time at the customer site.
xi. On-site service time is assumed to be φ minutes.
xii. Delivery time at the delivery location (depot or mail centre) should be no later than δ_D minutes after the specified delivery time.
xiii. Each unit of a resource can start out and return to the depot one or more times during the service sesssion.
xiv. When a unit resource starts out a second time from the depot, it can only visit a group of customer(s) whose latest pickup times and delivery times at the depot can be satisfied. (This applies to subsequent routes whenever the unit starts out from the depot.)
xv. When a unit resource 2 has collected its assigned customer documents, the customer location prior to returning to depot is a candidate site for documents transfer or picking up units of resource 1. Some (or all) resource 1 units may return together with resource 2; others may simply transfer its collected documents and continue to service other customers. Apart from these opportunities, no transportation of resource 1 units is considered to reduce risk of waiting and delay.
xvi. Transfer of collected documents or return together with resource 2 (in (xv)) is considered by checking two constraints: arrival time of a unit resource 1 at the transfer location (say customer i) is on or before the earliest pickup time (i.e., t_i - δ_P); the latest departure time at this transfer location (i.e., t_i + φ) still satisfies the delivery time window constraints at the depot (mail centre) for the collected documents.
xvii. The objective function of monthly total cost comprises the fixed cost and travelling cost of resource units, assuming a 5-day workweek and 4 weeks per month.

4. Modelling independent operations

It is common in vehicle routing and PDP literature to assume indepedent vehicle operations. The advantage of independence is the flexibility offered to vehicles to react to real-time changes without affecting other vehicles. Besides, the modelling approach and solution

methods are simpler than a cooperative strategy even for identical vehicles (Lin, 2008). An objective of this work is to examine possible cost savings and computational time required as delivery resources operate in some cooperative modes. Hence, the results from independent operations of the two resources can serve as a basis for comparison (Section 6).
Firstly, all independent customer groups that can be visited by each unit of a resource in a ***single route*** (starting and ending at depot) are found by enumerating all possible customer sequences in an enumeration tree (Brϕnmo et al., 2007). Let n_k be the number of feasible customer groups formed for resource k (k = 1, 2). A resource unit could be assigned multiple routes in a service session (assumption (xiii)) as it has to return documents to the depot to fulfil the delivery time window constraints, possibly before the session ends. The multi-route operations are represented by connected nodes in a network. Customer groups (sequence of customers visited in a single route) form nodes of this network, denoted by $\Pi = (\Psi, A)$, where Ψ and A represent the node set and arc set, respectively.

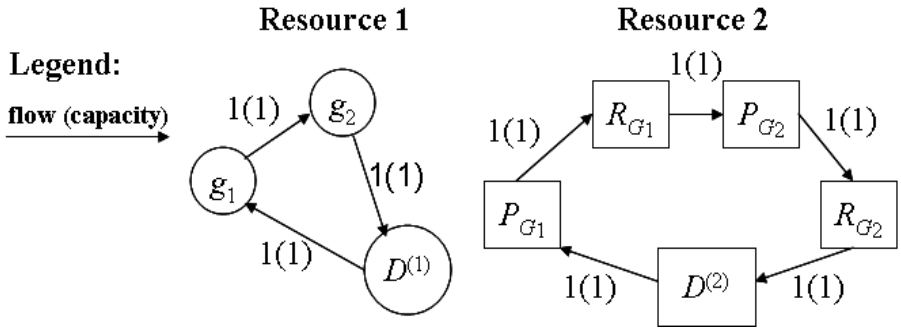


Fig. 1. Independent operations of two resource units

Figure 1 depicts instances of independent operations of a unit of each resource represented in Π. A unit resource 1 starts out from the depot (node $D^{(1)}$) to pick up group g_1 documents and returns to the depot (not shown in figure). Then it starts out again to pick up group g_2 documents and return to the depot (node $D^{(1)}$) to end its service. Similarly, a unit resource 2 first leaves the depot (node $D^{(2)}$) for group G_1 documents. On returning them to the depot (node R_{G1}), it starts out again to pick up group G_2 documents and return to the depot (node $D^{(2)}$) to end its service. Details of constructing network Π are explained as follows.
Node set Ψ: The single depot is represented by nodes $D^{(1)}$ and $D^{(2)}$ for resource 1 and 2, respectively. Each independent customer group, say g_i, of resource 1 forms a node itself (i = 1,..., n_1). Each independent customer group, say G_i, of resource 2 generates two nodes, P_{Gi}, and R_{Gi} (i = 1,..., n_2). Node P_{Gi} represents selection of customer group G_i to be serviced by a unit resource 2. Node R_{Gi} is a copy of the depot, to where resource 2 can return the collected documents of group G_i and start out again, if necessary. (The reason for adopting two nodes for each customer group of resource 2 is to allow future addition of cooperative arcs to be introduced in the subsequent sections. Hence, Π will be expanded systematically, aiming to obtain better solutions by cooperation between resource units.)
Arc set A : Each arc defined has the time constraints checked between its start node and end node. Each resource is associated with arcs defined on its own customer groups. Arc capacity is one unit for all arcs as there is no interaction between the two resources under independent operations. Seven types of arcs are defined in Π:

- ($D^{(1)}$, g): Group g is selected. A unit resource 1 starts out from the depot (node $D^{(1)}$) to collect documents in group g. (Arc cost is the sum of fixed cost of a unit resource 1 and the travelling cost servicing group g.)
- (g_i , g_j): Group g_i documents have been collected and returned to the depot by a unit resource 1. It starts out again from the depot to collect documents in group g_j ($i \neq j$). (The feasibility check for defining this arc ensures that even if the unit returns to the depot at the latest time of group g_i documents, it will not violate the pickup and delivery time windows of group g_j. This arc models multi-route assignment to each unit resource 1 within a service session. Arc cost is the travelling cost servicing group g_j from and back to the depot.)
- (g, $D^{(1)}$): Group g documents are returned to the depot and the unit resource 1 ends its service at node $D^{(1)}$ (depot). (Arc cost is 0.)
- ($D^{(2)}$, P_G): Group G is selected. A unit resource 2 starts out from the depot (node $D^{(2)}$) to collect documents in group G. (Arc cost is the fixed cost of a unit resource 2.)
- (P_G, R_G): Group G documents have been collected by a unit resource 2 which will return directly to the depot (node R_G) from the last customer location in G. (Arc cost is defined as the travelling cost servicing group G, from and back to the depot.)
- (R_{Gi}, P_{Gj}): Group G_i documents have been delivered to the depot (meeting the earliest delivery time constraint of Group G_i documents). A unit resource 2 start outs again to collect documents in group G_j ($\neq G_i$). The start time from the depot is determined by the earliest return time of Group G_i documents. (This arc models multi-route assignment to each unit resource 2 within a service session. Arc cost is 0.)
- (R_G, $D^{(2)}$): Group G documents are returned to the depot and the unit resource 2 ends its service at node $D^{(2)}$ (depot). (Arc cost is 0.)

Naturally, there is more than one way to model (independent) multi-route operations. An exact method includes enumerating all possible sequences of single routes, one after another, to form feasible sets of multi-route solutions for selection. However, this approach makes it difficult to model interaction and cooperation between different routes. The proposed network Π is a simpler representation. Nevertheless, some multi-route solutions have not been included due to assumption (xiv) made to simplify modelling. With the independent solutions represented by Π, the optimal solution can be obtained from an integer programming model:

Model P: Integer programming model for independent operations of two resources

Basic parameters

S_{kj} = set of customer groups of resource k (= 1, 2) that can service customer j (j = 1,…, N)

W_k = fixed cost per unit resource k (= 1, 2).

$\chi_g^{(1)}$ = travelling cost of a unit resource 1 in servicing customer group g, including the trips out of and return to the depot, g = 1,..., n_1

$\chi_G^{(2)}$ = travelling cost of a unit resource 2 in servicing customer group G, including the trips out of and return to the depot, G = 1,..., n_2

Decisions:

y_{ij} = flow (or connection) along arc (i, j), $\forall$ (i, j)$\in A$

$$Min.\ \ Z = \sum_{g=1}^{n_1} W_1 \cdot y_{D^{(1)},g} + \sum_{G=1}^{n_2} W_2 \cdot y_{D^{(2)},P_G} + \sum_{g=1}^{n_1} \sum_{(i,g)\in A} \chi_g^{(1)} \cdot y_{i,g} + \sum_{G=1}^{n_2} \chi_G^{(2)} \cdot y_{P_G R_G} \qquad (1)$$

subject to:

$$\sum_{g \in S_{1j}} \sum_{(i,g) \in A} y_{i,g} + \sum_{G \in S_{2j}} \sum_{(i,P_G) \in A} y_{i,P_G} = 1 \ , \ j = 1,\ldots, N \tag{2}$$

$$\sum_{(k,i) \in A} y_{k,i} = \sum_{(i,j) \in A} y_{i,j} \ , \ \forall \ i \in \Psi \tag{3}$$

$$y_{ij} = 0, 1, \forall (i, j) \in A \tag{4}$$

The objective function in constraint (1) is the (monthly) total cost, comprising the fixed costs of resources and travelling cost of servicing customer groups. Each customer must be visited exactly once in one of the customer groups. This is formulated in constraint (2). Constraint (3) describes the flow balance equation for each node in the network, with flow variables defined as binary integers in constraint (4). (In many instances, the integer variables of y_{ij} can be relaxed and the resulting optimal solution still maintains the integer property.)

5. Modelling transfer operations and return transporation

This section models the cooperative operations between units of the two resources by allowing documents transfer and/or return transportation of resource 1 units by resource 2. Such cooperation is represented by additional arcs embedded into network Π. There are certain rules defining the transfer operations and return transportation in this model, respectively. A unit resource 1 on collecting all documents in its assigned customer group can travel to transfer the collected documents to a unit resource 2 returning to the depot (from its last customer location), provided that it does not incur delay to the resource 2 unit and the delivery time constraints for both customer groups can be satisfied (assumption (xv)). After transfer, the resource 1 unit is free to service other customer groups from the transfer location. (Note that resource 2 stays dominant and its schedule would not be affected by document transfer or transporting others.) Alternatively, the resource 1 unit can return to the depot together with the resource 2 unit to save travelling cost. On its return to the depot, the resource 1 unit ends it service. To model the cooperations in this section, network Π (for modelling the independent operations)will be expanded by including additional nodes and arcs. The expanded network is denoted by $\Pi^T(\Psi^T, A^T)$, where Ψ^T and A^T represent the node set and arc set, respectively. Figure 2 depicts two types of cooperations between two resource units, to be represented in the network Π^T. In Figure 2(i), a unit resource 1 starts out from the depot (node $D^{(1)}$) to pick up documents of a customer group, denoted by g^T. This unit then travels to a transfer node T_1 which is the customer location of a resource 2 group G_1 prior to returning to depot. After documents transfer, the unit resource 1 continues to service another group g_2 before ending its service at the depot (node $D^{(1)}$), while the unit resource 2 returns with the collected and transferred documents (G_1 and g^T) to the depot (node $D^{(2)}$) and ends it service. Figure 2(ii) depicts the case of return transportation of a unit resource 1 by resource 2. A unit resource 1 starts out to collect group g documents, returns them to the depot and continues to service group g^T. After picking up group g^T documents, it joins a unit resource 2 at the last customer location of its assigned group G_2. Both return together to their respective depot (possibly the same physical location) and end their service.

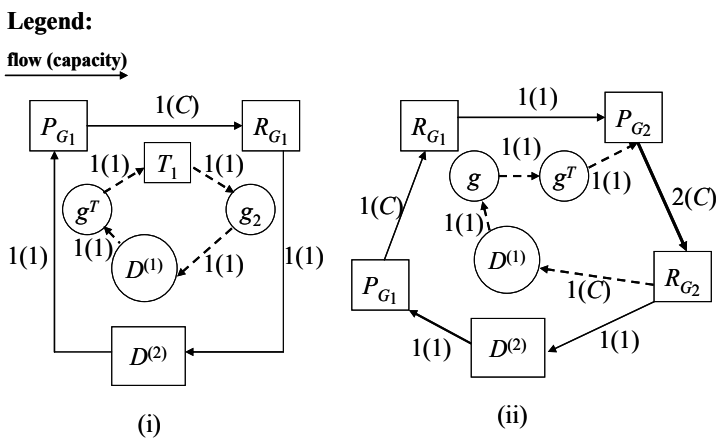


Fig. 2. Modelling cooperative operations between two resource units: (i) documents transfer and (ii) return transportation

Node set Ψ^T : In addition to nodes defined in Ψ, two types of new nodes will be included. The first type defines resource 1 transfer groups which represents the sequences of customers that can be visited by a unit resource 1 in a single route, starting from the depot and ending at a transfer node. The transfer node is selected at a customer site based on assumption (xvi). (The customer at this site is the last customer in a route for a resource 2 unit.) Transfer groups are found by enumerating all possible customer sequences as for the independent customer groups. As the return time by resource 2 (from the transfer node) is faster, more customers can be visited by a unit resource 1 in a single route before heading for transfer. Hence, there are more transfer groups than pure customer groups for resource 1. Let n_1^T be the number of transfer groups formed for resource 1. Each transfer group, say g_i^T , of resource 1 forms a node itself (i = 1,..., n_1^T) in network Π^T. The second type of new nodes is the set of transfer nodes. A total of N transfer nodes, T_1, T_2,..., T_N, are formed, representing customer location 1, 2,…, N respectively. (As a customer could belong to more than one customer group, each transfer node can be associated with one or more customer groups of resource 2 to be connected by arcs in network Π^T.)

Arc set A^T: In addition to arcs in A, new arcs are defined to model the cooperative operations in this section. Each new arc is checked for feasibility of time constraints between its start node and end node. Changes in paramters associated with arcs in A will be described.

- ($D^{(1)}$, g^T): Transfer group g^T is selected. A unit resource 1 starts out from the depot (node $D^{(1)}$) to collect documents in this group and travels to the last site for documents transfer to some resource 2 unit. (Arc capacity is one and arc cost is the sum of fixed cost of a unit resource 1 and the travelling cost servicing group g^T.)
- (g^T, P_G): This arc models return transportation. A unit resource 1, on collecting customer documents in transfer group g^T , joins a unit resource 2 at the last customer location (also a candidate transfer node) of its assigned group G. They return together to the depot in the same vehicle. (Arc capacity is one and arc cost is 0.)
- (P_G, R_G): Same interpretation as in arc set A, but arc capacity is changed to C (i.e., carrying capacity of resource 1 units, including the operator, by a unit resource 2).

- (R_G, $D^{(1)}$): This arc models return transportation. This new arc allows resource 1 unit(s) to be carried by a unit resource 2, responsible for Group G documents, to return to node $D^{(1)}$ (depot) and ends its service. (Arc capacity is C and arc cost is 0.)
- (g^T, T_i): This arc models pure document transfer. A unit resource 1, on collecting customer documents in transfer group g^T, transfers its documents to a unit resource 2 servicing customer i (transfer location), before its return to depot. (Note that customer i is stored as the last element in group g^T during the enumeration procedure. Arc capacity is one and arc cost is 0.)
- (T_i , g): This arc models assignment after document transfer. It allows a unit resource 1 to service customer group g after it is released at customer location i, the transfer location. (Arc capacity is one. Arc cost is the travelling cost servicing group g by starting at location i and ending at the depot.)
- (T_i , g^T): Similar interpretation and parameters as for arc type (T_i , g), but the customer group to be serviced is transfer group g^T instead.
- (T_i , $D^{(1)}$): This arc models the end of service after document transfer. It represents a unit resource 1 returning to node $D^{(1)}$ (depot) to end its service, from the transfer location at customer i. (Arc capacity is one and arc cost is the travelling cost from customer i to the depot.)
- (g_i , g^T): Similar to arc type (g_i, g_j) in arc set A, this arc models the multi-route assignment of a unit resource 1. Group g_i documents are returned to the depot and the unit resource 1 starts out again from the depot to collect documents in transfer group g^T. (Arc capacity is one and arc cost is the travelling cost servicing transfer group g^T, including the trips from the depot and to the transfer location.)

With the network Π^T expanded from Π to model the cooperative operations, the optimal solution can be obtained from a mixed integer programming model extended from Model P with side constraints:

Model Γ: Mixed integer programming model with transfer opportunity and return transportation

Additional parameters

S^T_{1j} = set of transfer groups of resource 1 that can service customer j (j = 1,…, N)

L_{2j} = set of customer groups of resource 2 that service customer j last (j = 1,…, N)

$\chi^{(1)}_{g^T}$ = travelling cost of a unit resource 1 in servicing transfer group g^T, including the outbound trip from depot and the trip to the transfer location, the last element stored in g^T (g^T = 1,..., n_1^T)

Decisions:

y_{ij} = flow (or connection) along arc (i, j), $\forall$ (i, j)$\in A^T$

$$\delta_{P_G,R_G} = \begin{cases} 1 & \text{if group } G \text{ is serviced by a unit resource 2,} \\ 0 & \text{otherwise} \end{cases} \qquad G=1,..., n_2$$

$$\begin{aligned} Min.\ \ Z = & \sum_{g=1}^{n_1} W_1 \cdot y_{D^{(1)},g} + \sum_{G=1}^{n_2} W_2 \cdot y_{D^{(2)},P_G} + \sum_{g^T=1}^{n_1^T} W_1 \cdot y_{D^{(1)},g^T} + \\ & \sum_{g=1}^{n_1} \sum_{(i,g)\in A^T} \chi_g^{(1)} \cdot y_{i,g} + \sum_{G=1}^{n_2} \chi_G^{(2)} \cdot \delta_{P_G R_G} + \sum_{g^T=1}^{n_1^T} \sum_{(i,g^T)\in A^T} \chi_{g^T}^{(1)} \cdot y_{i,g^T} \end{aligned} \tag{5}$$

subject to:

$$\sum_{g \in S_{1j}} \sum_{(i,g) \in A^T} y_{i,g} + \sum_{G \in S_{2j}} \delta_{P_G,R_G} + \sum_{g^T \in S_{1j}^T} \sum_{(i,g^T) \in A^T} y_{i,g^T} = 1 \ , \ j = 1,\ldots, N \tag{6}$$

$$\sum_{(k,i) \in A^T} y_{k,i} = \sum_{(i,j) \in A^T} y_{i,j} \ , \ \forall \ i \in \Psi^T \tag{7}$$

$$y_{D^{(2)},P_G} + \sum_{(R_{G_i},P_G) \in A^T} y_{R_{G_i},P_G} = \delta_{P_G,R_G} \ , \ \forall \ (P_G, R_G) \in A^T \tag{8}$$

$$\sum_{(g^T,P_G) \in A^T} y_{g^T,P_G} \le (C-1) \cdot \delta_{P_G,R_G} \ , \ \forall \ (P_G, R_G) \in A^T \tag{9}$$

$$\sum_{(i,T_j) \in A^T} y_{i,T_j} \le (N-1) \cdot \sum_{G \in L_{2j}} \delta_{P_G,R_G} \ , \ j = 1,\ldots, N \tag{10}$$

$$\delta_{P_G,R_G} = 0, 1; \ 0 \le y_{P_G,R_G}, y_{R_G,D^{(1)}} \le C \ \ \forall \ (P_G, R_G), (R_G, D^{(1)}) \in A^T \tag{11}$$

$$y_{ij} = 0, 1, \forall (i, j) \in A^T \setminus \{ (P_G, R_G), (R_G, D^{(1)}) \} \tag{12}$$

The objective function in constraint (5), the demand constraint (6) for each customer and the flow balance constraint (7) for each node in network Π^T is modified from constraint (1), (2) and (3) in Model P, respectively. (For demand constraint (6), only customer locations in transfer group g^T will be considered, but not its last element (transfer location).) Constraints (8) to (10) are the side constraints. Constraint (8) models the relationship between the selection decision (δ_{P_G,R_G}) of a resource 2 customer group (G) and the inflow arcs (of resource 2) to its associated node (P_G). Constraint (9) formulates the capacity constraint of resource 2 in carrying resource 1 units, before returning to the depot on finishing group G. Constraint (10) imposes condition on the use of transfer nodes (T_j). No document transfer can take place (at customer location j) if there is no resource 2 units servicing this customer last in its route (i.e., no customer groups in L_{2j} are selected). The last two constraints (11) and (12) declare the decision variables and their relevant bounds. (In many instances, some binary integer variables can be relaxed without affecting the integer property of the optimal solution.)

6. Computational experiments

Computational analysis is carried out based on two types of data. The first set (12 instances) contains customer locations obtained from real-life data of a local delivery service. Each instance contains between 27 to 30 customers. The second set (45 instances) are simulated test problems containing 50, 100 or 150 customers. Data and parameters were obtained from the following sources:

- From the local delivery service, the pairwise travel distances were provided either by the driver or estimated by geographical information systems (GIS), given the customer and depot locations. The unit travelling cost of resource 2 (vehicle) is HK$70 per hour.
- From a courier service, the duration of a service sesstion (T) is 300 minutes; the on-site service time (φ) is 5 minutes; the early allowance (δ_P) for the pickup time window is 5 minutes; the lateness allowance (δ_D) for the delivery time window is 10 minutes. The monthly fixed cost per unit resource 1 (W_1) and resource 2 (W_2) is HK$7,000 and HK$19,000, respectively. The carrying capacity (C) of resource 1 units by a unit resource 2 is 6.
- The travelling speed of resource 2 (V_2) is based on the average vehicle travel speed of 20.8 km/hour from a local study (Transport Department, Hong Kong, 2001).
- The travelling speed of resource 1 (V_1) is obtained from the average human walking speed of 5 km/hour from Wikipedia, the free online encyclopedia.

Unknown parameters of pickup times (t_i) and delivery times (τ_i) for all instances are simulated within the time interval [0, T]. For the simulated instances, customer locations are randomly generated in a rectangular travel grid of size 200×200 min^2, such that the maximum one-way travel time is around 280 minutes. This is closed to the duration of service session (T) and is reasonable for the size of urban cities. For the simulated instances, the depot locations are systematically set at 3 positions in the travel grid: the edge, the centre and half-way between, to examine the differences in solutions.

The two models (independent Model P and cooperative Model Γ) were coded in Visual Basic.NET 2005 version and all test instances are solved by the optimization software CPLEX 10.1. All experiments were performed on a Pentium 4, 2.5 GHz processor. Comparison between the two models are based on the performance measures of (monthly) total cost, units of each resource required and the computational time taken. Detailed results on cost and resources requirement are recorded in Table 1 and Table 2 for the two types of data respectively. Computational time for all instances are given in Table 3.

An example of optimal solution is shown in Figure 3. A unit resource 2 is assigned two routes: 0 → 1 → 2 → 0 → 3 →0 while a unit resource 1 is assigned a single route to service customer node 4. After collecting documents at node 4, it travels to node 2 for a free ride back to node 0 (depot) provided by resource 2.

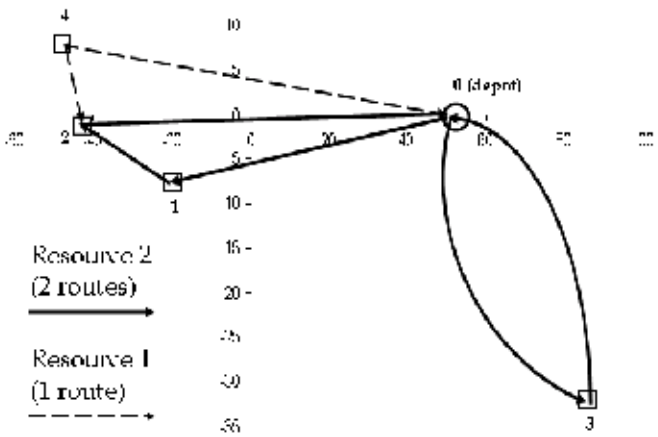


Fig. 3. Optimal cooperation: Return transportation of unit resource 1 by resource 2 (pickup at customer node 2 and jointly return to depot)

Instance	No. of customers	Monthly total cost in HK$ (units required of resource 2, resource 1 respectively)	
		Independent operations (Model *P*)	Transfer, return transport (Model Γ)
1	27	308,320 (13, 1)	308,320 (13, 1)
2		299,803.33 (13, 0)	299,803.33 (13, 0)
3		270,740 (11, 1)	270,740 (11, 1)
4		319,106.67 (14, 0)	319,106.67 (14, 0)
5	29	139,433.33 (6, 0)	139,433.33 (6, 0)
6		193,836.67 (8, 0)	193,836.67 (8, 0)
7		337,983.33 (14, 0)	337,983.33 (14, 0)
8		202,453.33 (9, 0)	193,230 (8, 1)
9		161,210 (7, 0)	161,210 (7, 0)
10	30	188,493.33 (8, 0)	188,493.33 (8, 0)
11		230,693.33 (10, 0)	217,690 (9, 1)
12		261,313.33 (11, 0)	248,916.67 (10, 1)
Average cost savings (%) over independent operations [min.%, max.%]		-	1.24% [0%, 5.64%]

Table 1. Computational results for the local instances 1-12

Instance	No. of customers	Depot location	Monthly total cost in HK$ (units required of resource 2, resource 1 respectively)	
			Independent operations (Model *P*)	Transfer, return transport (Model Γ)
13	50	Edge (100, 0)	575,803.33 (22, 0)	571,876.67 (21, 2)
14			507,416.67 (19, 0)	507,416.67 (19, 0)
15			549,826.67 (21, 1)	549,593.33 (21, 1)
16			667,530 (26, 0)	663,743.33 (25, 2)
17			574,380 (22, 0)	570,010 (21, 2)
18		Half-way (50, 0)	563,326.67 (23, 0)	550,416.67 (22, 1)
19			522,783.33 (21, 0)	522,783.33 (21, 0)
20			461,753.33 (19, 0)	461,753.33 (19, 0)
21			528,016.67 (22, 0)	515,083.33 (21, 1)
22			511,746.67 (21, 0)	511,746.67 (21, 0)
23		Centre (0, 0)	507,383.33 (21, 0)	507,383.33 (21, 0)
24			511,630 (21, 0)	511,630 (21, 0)

25			547,296.67 (23, 0)	534,876.67 (22, 1)
26			529,136.67 (22, 0)	519,890 (21, 1)
27			510,656.67 (22, 0)	510,656.67 (22, 0)
Average cost savings (%) over independent operations [min.%, max.%]			-	0.72% [0%, 2.45%]
28	100	Edge (100, 0)	997,333.33 (38, 0)	962,616.67 (35, 3)
29			947,153.33 (36, 1)	929,563.33 (33, 6)
30			975,066.67 (37, 2)	927,443.33 (30, 14)
31			982,213.33 (38, 0)	982,213.33 (38, 0)
32			929,420 (36, 1)	902,940 (33, 5)
33		Half-way (50, 0)	833,150 (33, 3)	833,150 (33, 3)
34			869,890 (35, 0)	869,890 (35, 0)
35			971,370 (39, 1)	968,120 (38, 3)
36			859,460 (35, 0)	835,080 (32, 4)
37			728,590 (29, 1)	722,183.33 (27, 5)
38		Centre (0, 0)	762,926.67 (31, 1)	761,310 (30, 3)
39			813,993.33 (33, 1)	813,053.33 (32, 3)
40			761,853.33 (31, 0)	761,853.33 (31, 0)
41			887,473.33 (37, 0)	862,913.33 (35, 2)
42			887,590 (37, 0)	864,943.33 (35, 2)
Average cost savings (%) over independent operations [min.%, max.%]			-	1.52% [0%, 4.88%]
43	150	Edge (100, 0)	1,200,736.67 (46, 0)	1,172,693.33 (43, 4)
44			1,148,216.67 (43, 2)	1,123,656.67 (41, 4)
45			1,087,723.33 (41, 2)	1,071,773.33 (39, 5)
46			1,232,090 (45, 4)	1,209,590 (44, 4)
47			1,114,983.33 (42, 1)	1,111,570 (41, 3)
48		Half-way (50, 0)	1,152,856.67 (46, 1)	1,135,343.33 (44, 4)
49			1,034,913.33 (40, 3)	1,034,073.33 (40, 3)
50			1,045,536.67 (41, 2)	1,034,906.67 (39, 6)
51			1,052,146.67 (42, 0)	1,043,203.33 (41, 2)
52			1,291,133.33 (50, 2)	1,252,816.67 (46, 6)
53		Centre (0, 0)	1,128,910 (45, 2)	1,128,186.67 (45, 2)
54			1,076,216.67 (44, 0)	1,071,123.33 (43, 2)
55			1,089,400 (44, 1)	1,088,996.67 (43, 3)
56			1,152,490 (47, 1)	1,140,373.33 (46, 2)
57			1,063,540 (43, 1)	1,056,533.33 (42, 3)
Average cost savings (%) over independent operations [min.%, max.%]			-	1.12% [0.04%, 2.97%]

Table 2. Computational results for the simulated instances 13-57

The observations from the computational experiments are summarized as follows:

- Both models could be solved to optimality for all 57 instances within reasonable time. The maximum running time is within 1,200 CPU seconds for instances of up to 150 customers.
- In all instances, the cooperative model (Model Γ) allowing documents transfer and return transporation performs at least as good as the independent model (Model *P*). The percentage of cost savings could be as much as 5%, with an average between 1-2%. (As the network Π^T is expanded from the network Π of the independent model, the solution quality cannot be worse. Surprisingly, the computational time taken for both models are quick for instances of up to 150 customers.)
- The travel speed of resource 1 (couriers on foot) is relatively slower than resource 2 (vans). Return transportation by resource 2 is found more frequent when cooperative operations do occur. The number of resource 1 units used is small and they only serve a few customers (resulting in low resource utilization), as compared with resource 2 units.
- Some system characteristics will favor cooperative operations. When problem size increases or when the depot location is away from the customer centroid (say half-way, or at the edge of the service region), the optimal solution contains more cooperative operations, in particular, return transportation of resource 1 by resource 2.

There are naturally limitations of this work that need to be pointed out. Real data could contain clusters of nearby customer locations that would make it difficult to enumerate all customer groups. Some clustering of service areas could be carried out before applying any model. Scheduling and routing problems are often affected by factors that are not easily formulated by optimization models (e.g., personal preferences, fairness of assignment). Other complex factors in routing include the use of public transport, traffic congestion, turn restrictions on streets (Irnich, 2008) would require heuristics and communication technologies for real-time monitoring and control.

7. Conclusion

This work is a continuation of an earlier work on routing courier services (Lin, 2008). It contributes to studies on multi-resource scheduling in pickup and delivery operations by modelling certain cooperative operations. The static-deterministic problem could be formulated by a (mixed) integer programming model and for the 57 instances generated from real data and simulated data (consisting of up to 150 customers), the computational time is within 1,200 CPU seconds. With the slower resource as courier and the faster resource as van, return transportation of the slower unit(s) by a faster unit is found to generate cost savings for certain data sets (e.g., Figure 3). Monthly cost savings were achieved at an average of 1-2% (maximum of 5%) over the independent operations. For the courier industry which has experienced considerable growth in South-east Asia, a few percent of monthly total cost could represent large dollar savings in the long-run. As fuel cost rises, saving in travelling cost is expected to further increase. The optimization component could be further incorporated into intelligent transportation systems with route guidance system to improve the operational efficiency (Jung et al., 2006). Further interesting research directions include modelling the outbound transportation of the lighter resource by the heavier resource and developing heuristics for solving large problems efficiently, when the IP model requires large computational resource.

This work contributes to modelling cooperation and developing methods to solve hard delivery problems in real situations. In actual practice of courier companies, some cooperative strategies are in action but without sound theory to justify performance or to explore further improvement.

Instance	No. of customers	CPU seconds	
		Independent operations (Model *P*)	Transfer, return transport (Model Γ)
1	27	4.62	4.6
2		5.28	4.57
3		6.1	4.46
4		4.16	6.81
5	29	25.2	11.85
6		13.33	7.29
7		6.41	6.77
8		10.79	8.89
9		10.41	173.08
10	30	9.64	12.94
11		7.38	13.26
12		8.45	90.59

Instance	No. of customers	Depot location	CPU seconds	
			Independent operations (Model *P*)	Transfer, return transport (Model Γ)
13	50	Edge (100, 0)	11.33	10.97
14			12.41	12.77
15			12.59	12.07
16			10.83	12.65
17			12.18	11.53
18		Half-way (50, 0)	14.46	10.71
19			11.31	11.67
20			12.1	11.85
21			10.73	11.88
22			12.56	12.11
23		Centre (0, 0)	14.28	12.73
24			15.34	12.36
25			17.6	11.64
26			18.35	12.22
27			11.75	11.52

28	100	Edge (100, 0)	39.11	38.82
29			45.67	40.75
30			39.17	39.24
31			38.03	43.53
32			38.79	46.71
33		Half-way (50, 0)	40	42.56
34			42.39	85.17
35			46.54	99.98
36			42.35	52.55
37			41.7	46.67
38		Centre (0,0)	41.53	46.03
39			41.82	51.15
40			47.01	45.72
41			39.53	43.12
42			43.22	38.8
43	150	Edge (100, 0)	102.3	149.22
44			135.8	1,079.25
45			110.93	537.2
46			99.72	110.08
47			121.64	303.17
48		Half-way (50, 0)	138.21	1,019.17
49			129.43	714.23
50			566.5	629.97
51			142.65	440.06
52			119.13	196.46
53		Centre (0, 0)	88.36	179.43
54			139.76	717.35
55			105.91	299.05
56			91.06	119.78
57			119.72	478.24

Table 3. Computational time of instances 1-57 (local: 1-12; simulated: 13-57)

8. References

Brϕnmo, G.; Christiansen, M. & Nygreen, B. (2007). Ship routing and scheduling with flexible cargo sizes. *Journal of the Operational Research Society*, Vol. 58, No. 9, September 2007, 1167-1177, ISSN: 01605682

Chao, I.-M. (2002). A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, Vol. 29, No. 1, January 2002, 33-51, ISSN: 0305-0548

Irnich, S. (2008). Solution of real-world postman problems. *European Journal of Operational Research*, Vol. 190, No. 1, October 2008, 52-67, ISSN: 0377-2217

Jung, H.; Lee, K. & Chun, W. (2006). Integration of GIS, GPS, and optimization technologies for the effective control of parcel delivery service, *Computers & Industrial Engineering*, Vol. 52, No. 4, September 2006, 154-162, ISSN: 0360-8352

Kamoun, M. & Hall, R.W. (1996). Design of express mail services for metropolitan regions. *Journal of Business Logistics*, Vol. 17, No. 2, 1996, 265-302, ISSN: 0735-3766

Langevin, A. & Soumis, F. (1989). Design of multiple-vehicle delivery tours satisfying time constraints. *Transportation Research Part B*, Vol. 23, No. 2, April 1989, 123-138, ISSN: 0191-2615

Lin, C.K.Y. (2008). A cooperative strategy for a vehicle routing problem with pickup and delivery time windows. *Computers & Industrial Engineering*, in press, ISSN: 0360-8352

Lu, Q. & Dessouky, M.M. (2006). A new insertion-based construction heuristic for solving pickup and delivery problem with time windows. *European Journal of Operational Research*, Vol. 175, No. 2, December 2006, 672-687, ISSN: 0377-2217

Mitrović-Minić, S.; Krishnamurti, S., R. & Laporte, G. (2004). Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B*, Vol. 38, No. 8, September 2004, 669-685, ISS: 0191-2615

Mitrović-Minić, S. & Laporte, G. (2004). Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B*, Vol. 38, No. 7, September 2004, 635-655, ISSN: 0191-2615

Mitrovic-Minic, S. & Laporte, G. (2006). The pickup and delivery problem with time windows and transshipment. *INFOR*, Vol. 44, No. 3, August 2006, 217-227, ISSN: 0315-5986

Ropke, S. & Pisinger, D. (2006). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, Vol. 171, No. 3, June 2006, 750-775, ISSN: 0377-2217

Savelsbergh, M.W.P. & Sol, M. (1995). The general pickup and delivery problem. *Transportation Science*, Vol. 29, No. 1, February 1995, 17–29, ISSN: 0041-1655

Scheuerer, S. (2006). A tabu search heuristic for the truck and trailer routing problem. *Computers & Operations Research*, Vol. 33, No. 4, April 2006, 894-909, ISSN: 0305-0548

Semet, F. (1995). A two-phase algorithm for the partial accessibility constrained vehicle routing problem. *Annals of Operations Research*, Vol. 61, No. 1-4, December 1995, 45-65, ISSN: 0254-5330

Shang, J.S. & Cuff, C.K. (1996). Multicriteria pickup and delivery problem with transfer opportunity. *Computers & Industrial Engineering*, Vol. 30, No. 4, September 1996, 631-645, ISSN: 0360-8352

Transport Department, Hong Kong. (2001). Car journey time survey for monitoring traffic congestion, Hong Kong.

Yu, V.F.; Lin, S.-W. & Chou, S.-Y. (2008). Solving the truck and trailer routing problem based on a simulated annealing heuristic. *Computers & Operations Research*, in press, ISSN: 0305-0548

Edited by Tonci Caric and Hrvoje Gold

The Vehicle Routing Problem (VRP) dates back to the end of the fifties of the last century when Dantzig and Ramser set the mathematical programming formulation and algorithmic approach to solve the problem of delivering gasoline to service stations. Since then the interest in VRP evolved from a small group of mathematicians to a broad range of researchers and practitioners from different disciplines who are involved in this field today. Nine chapters of this book present recent improvements, innovative ideas and concepts regarding the vehicle routing problem. It will be of interest to students, researchers and practitioners with knowledge of the main methods for the solution of the combinatorial optimization problems.

ISBN 978-953-51-6401-2

Photo by FooTToo / iStock

IntechOpen