IntechOpen

Particle Swarm Optimization with Applications

Edited by Pakize Erdoğmuş

PARTICLE SWARM OPTIMIZATION WITH APPLICATIONS

Edited by **Pakize Erdoğmuş**

Particle Swarm Optimization with Applications
http://dx.doi.org/10.5772/intechopen.69826
Edited by Pakize Erdoğmuş

Contributors

Arunachalam Sundaram, Mostafa Kheshti, Lei Ding, Alma Y. Alanis, Eduardo Rangel, Esteban A. Hernandez-Vargas, Nancy Arana-Daniel, Carlos Lopez-Franco, Subhrapratim Nath, Ruey-Maw Chen, Yin-Mou Shen, Pakize Erdogmus

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 London, United Kingdom, 2018 by IntechOpen
eBook (PDF) Published by IntechOpen, 2019
IntechOpen is the global imprint of INTECHOPEN LIMITED, registered in England and Wales, registration number: 11086078, The Shard, 25th floor, 32 London Bridge Street
London, SE19SG – United Kingdom
Printed in Croatia

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

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

Particle Swarm Optimization with Applications
Edited by Pakize Erdoğmuş

p. cm.

Print ISBN 978-1-78923-148-9

Online ISBN 978-1-78923-149-6

eBook (PDF) ISBN 978-1-83881-430-4

Meet the editor

Pakize Erdoğmuş was born in Erzurum, Turkey, in 1972. She received her BSc degree in Electronics and Communications Engineering from Yildiz Technical University, Kocaeli Engineering Faculty, in 1993. She also received her MSc degree in Computer Science and her PhD degree in Numerical Methods. She studied continuous optimization problems, linear programming and revised simplex method, for her master of thesis and course scheduling "Discrete Optimization Problem" for her PhD.

From 2003 to 2010, she was an Assistant Professor in Duzce University, Faculty of Technical Education. Since 2010, she has been working in Department of Computer Engineering of Duzce University, Faculty of Engineering. She is the author of more than 20 articles. Her research interests are nature-inspired optimization algorithms, signal and image processing.

Contents

Preface

Mankind lives in an environment where resources are limited. Therefore, since the day he existed, he has been trying to find the best solution to the problems he has encountered using these limited resources.

Like most of the inventions, optimization studies took a long way in the World War II. Linear programming was developed as a discipline in the 1940s, motivated initially by the need to solve complex planning problems in wartime operations.

With the developments of computers and computer programming, complex optimization problems encountered in the industry have been solved scientifically through modelling and optimization. These efforts have used classical mathematical solution methods and algorithms. But it has been seen that the solution time has been increased exponentially with the number of independent variables of an optimization problem.

Instead of finding the exact solution of a given problem in a long time, finding near-optimal solution of a given problem in a reasonable solution time has been accepted. So, this type of solution simulates animal behaviours and intelligence. Since most creatures behave optimally using their swarm intelligence, scientific studies on optimization have turned their way to these algorithms. Starting with ant colony optimization and particle swarm optimization, optimization algorithms simulating swarm intelligence have increased very much. Today, nearly all living beings' behaviours and their intelligence have been simulated for solving the nonlinear optimization problems.

The intent of this book is to give readers some insights into particle swarm optimization (PSO), which is one of the most used and cited nature-inspired optimization algorithms. PSO developed by Kennedy and Eberhart is both fast convergent and a very simple algorithm requiring very few parameters. So, since 1995, PSO has been popular and preferred especially in the dynamic optimization. Several hybrid versions of PSO have been improved.

This book presents some real-world applications of PSO. We intend to give a perspective to the readers who are planning to use PSO to solve a real-life problem.

Chapter 1 briefs the origins of PSO, improvements in the algorithm, hybrid studies and applications in the literature.

PSO with a bio-inspired aging model is introduced in Chapter 2. This model is proposed for alleviating the premature convergence problem.

"PSO Solution for Power System Operation Problems" is presented in Chapter 3. Two case studies were handled. The first case study system investigates applicability of PSO on pro-

viding proper overcurrent relay settings in the grid. The second case study system, the economic dispatch of a 15-unit system, is solved with PSO.

In Chapter 4, "Stochastic Greedy-Based PSO for Workflow Application in Grid" is presented.

In Chapter 5, "Performance Comparison of PSO and Its New Variants in the Context of VLSI Global Routing" is presented.

In Chapter 6, one of the most studied problems in electrical engineering "Combined Economic Emission Dispatch Problem with Valve-Point Effect Using Hybrid NSGA II-MOPSO" is presented.

I hope this book will be helpful to the readers.

Pakize Erdoğmuş
Duzce University
Turkey

Chapter 1

Introductory Chapter: Swarm Intelligence and Particle Swarm Optimization

Pakize Erdogmus

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.74076

1. Introduction

In order to survive, the main objective of all creatures is foraging. Foraging behavior is cooperative in the same species. Each agent in the swarm communicates with others in such a way to find the food in the shortest time and way. This capability of all lively beings gives inspiration to the human being in order to find solutions to the optimization problems. Collective foraging behaviors of the lively beings are called swarm intelligence.

Most of the animals live as social groups in order to find foods easily and protect from the enemies to survive. Each individual lives in their habitat. Looking for food, they use their own experiences called cognitive movements as well as the experience of their leaders called social movements.

Optimization is to find the best solution to a given problem under some constraints. All disciplines use optimization for finding the best solution for their problems. Optimization is the first and foremost objective for engineers too. So especially in the future engineering applications, optimization will be an indispensable part of the product.

Optimization is everywhere. In the production of a new device, in a new artificial intelligence technique, in a big data application or in a deep learning network, optimization is the most important part of the application. To design a device with optimum sizes using minimum energy, to train a network, to minimize the error between the desired output and real output values, optimization is required.

Because of the difficulties of classical optimization algorithms, scientists have started to find an easy way to solve their problems in the last 1960s. The development of the computers made the efforts of the scientists easy, and completely new problem solution techniques are

studied. These techniques using heuristic information were derivative free, easy to implement, and shorten the solution time. The first product of these studies is genetic algorithm (GA) developed by Holland [1]. The evolutionary idea has been applied to the solution of the optimization problems. Instead of the evolving only one solution, a group of solutions called population has been used in the algorithm. Each solution is called individual. By this way, running such algorithms with multiple processors could be possible. After GA, simulated annealing [2] has been generally accepted as the second algorithm, inspired from the annealing process of physical materials. In high temperatures, particles move randomly in order to explore the solution space. While temperature is decreasing, particles try to create a perfect crystalline structure, only with local movements.

2. Particle swarm optimization

Particle swarm optimization (PSO) is accepted as the second population-based algorithm inspired from animals. Since James Kennedy (a social psychologist) and Russell C. Eberhart simulated the bird flocking and fish schooling foraging behaviors, they have used this simulation to the solution of an optimization problem and published their idea in a conference in 1995 [3] for the optimization of continuous nonlinear functions. There are two main concepts in the algorithm: velocity and coordinate for each particle. Each particle has a coordinate and an initial velocity in a solution space. As the algorithm progresses, the particles converge toward the best solution coordinates. Since PSO is quite simple to implement, it requires less memory and has no operator. Due to this simplicity, PSO is also a fast algorithm. Different versions of PSO have been developed, using some operators since the first version of PSO was published.

In the first versions of PSO, the velocity was calculated with a basic formula using current velocity, personal best and local best values in the formula, multiplying stochastic variables. The current particle updates its previous velocity, not only its previous best but also the global best. The total probability was distributed between local and global best using stochastic variables.

In the next versions, in order to control the velocity, an inertia weight was introduced by Shi and Eberhart in 1998 [4]. Inertia weight balances the local and global search ability of algorithm. Inertia weight specifies the rate of contribution of previous velocity to its current velocity. Researchers made different contributions to the inertia weight concept. Linearly, exponential or randomly decreasing or adaptive inertia weight was introduced by different researchers [5]. In the next version of PSO, a new parameter called constriction factor was introduced by Clerc and Kenedy [6, 7]. Constriction factor (K) was introduced in the studies on stability and convergence of PSO. Clerc indicates that the use of a constriction factor insured convergence of the PSO. A comparison between inertia weight and constriction factor was published by Shi and Eberhart [8].

Nearly all engineering discipline and science problems have been solved with PSO. Some of the most studied problems solved with PSO are from Electrical Engineering, Computer Sciences, Industrial Engineering, Biomedical Engineering, Mechanical Engineering and Robotics. In Electrical Engineering, power distribution problem [9] is solved with PSO. Another most studied problem in Electrical Engineering is economic dispatch problem [10, 11]. In Computer Sciences, face localization [12], edge detection [13], image segmentation [14], image denoising

[15], image filtering [16] problems are solved with PSO. In Industrial Engineering, examination timetabling problems [17], traveling salesman problem [18], and job-shop scheduling problems [19] are solved with PSO. In Robotics, particle swarm optimization in coconut tree plucking robot is introduced [20], and path planning problem is solved with PSO [21]. In these studies, it has been proven PSO success in point of both performance and speed in most of the studies.

After the main PSO algorithm was studied and evolved with some parameters, hybrid algorithms were designed and developed by researchers. The most prominent ones are hybrid PSO with genetic algorithm (GA) [22] and particle swarm optimization with chaos [23–25] and quantum chaotic PSO [26]. In the next section, some of the recent hybrid PSO algorithms are presented.

3. Hybrid PSO algorithms using swarm intelligence

Hybrid algorithms are quite successful since they combine both algorithms' powerful sides. Since PSO is quite fast algorithm, nearly all newly developed algorithm combined with PSO. Some of the recent studies using swarm intelligence are crow search algorithm (CSA) [27], ant lion optimizer (ALO) [28], the whale optimization algorithm (WOA) [29], grey wolf optimizer (GWO) [30], monarch butterfly optimization (MBO) [31], moth flame optimization [32], selfish herd optimization (SHO) [33], and salp swarm optimization (SSO) [34]. Since the algorithms stated in the following paragraph are quite new, according to the Web of Science records, there is not any hybrid study combining PSO with them.

Firefly algorithm (FFA) [35], bacterial foraging optimization algorithm (BFOA) [36], ant colony optimization (ACO) [37], artificial bee colony (ABC) [38], and cuckoo search (CS) [39] are some of the algorithms using swarm intelligence improved in the last decade. There are hybrid versions of these algorithms with PSO.

3.1. Firefly algorithm

FFA is improved by mimicking the flashing activity of fireflies. FFA is similar most of the swarm intelligence algorithm. Fireflies are located in a position in the solution space randomly initially. The fitness of the fireflies is calculated according to the light intensity. The next location of each firefly is calculated according to the current position, randomness and attractiveness. The hybrid algorithm combined PSO with firefly optimization [40] proposes a technique for the detection of Bundle Branch Block (BBB), one of the abnormal cardiac beat, using hybrid firefly and particle swarm optimization (FFPSO) technique in combination with Levenberg Marquardt Neural Network (LMNN) classifier.

3.2. Bacterial foraging optimization algorithm

BFOA is inspired by the social foraging behavior of *Escherichia coli*. BFOA is an efficient algorithm in solving real-world optimization problems. Chemotaxis process simulates the movement of an *E. coli* cell through swimming and tumbling via flagella. *E. coli* cells like particles move in solution space and change their location with a formula-dependent previous

location, randomness and chemotactic step size. The big difference between PSO is that not only global best, but also global worst is also evaluated in the algorithm. Among *E. coli* cells, the least healthy one, eventually die. Each of the healthier bacteria splits into two bacteria, which are then placed in the same location. This keeps the swarm size constant. The hybrid algorithm using PSO and BFOA optimized PI controller, for multiobjective load frequency [41]. The authors developed a hybrid PSO with firefly, also developed a hybrid PSO with BFOA, for the detection of BBB. The same classifier is used in the study [42].

3.3. Ant colony optimization

ACO is inspired from the pheromone trails of ants. The first version improved by Dorigo is called ant system [43]. Although most of the ant species are blind, they can find the shortest path from their nest to food source using swarm intelligence. Ants are located in random positions in the solution space and moves with pheromone trail and randomness. In the first iterations, ants move stochastically. Eventually, pheromone increases in the path used most because ants prefer the path that contains more pheromone. This means that "Trace me." In order not to get trapped to the local convergence, pheromone evaporates related to time. ACO is generally used for the solution of combinatorial optimization problem solution such as travelling salesman problem (TSP) and some network problems. Hybrid PSO with ACO [44] solves routing problem.

3.4. Artificial bee colony

ABC optimization is developed by Karaboga in 2005. ABC is also a swarm intelligence algorithm based on the foraging behavior of honey bee swarms. The artificial bee colonies in the ABC algorithm consists of three groups: employed bees, onlookers and scouts. Employed bees search for food source and sharing this information to recruit onlooker bees. Onlooker bees select better food sources from those employed bees and further search around the selected food source. If a food source is not improved by some iteration, this employed bee will become a scout bee to search randomly for new food sources. In [45], a hybrid algorithm is developed combining PSO and ABC. Since PSO is fast convergent algorithm and ABC is slow convergent algorithm, hybrid algorithm uses the powerful sides of each algorithm.

3.5. Cuckoo search

CS is also another swarm intelligence algorithm inspired from cuckoos. CS is based on the interesting breeding behavior such as brood parasitism of certain species of cuckoos in combination with L'evy flight behavior of some birds. CS is successful for finding optimum values of multimodal functions. A hybrid algorithm is proposed finding for optimal design of multiband stop filters [46].

4. Conclusion

After GA, PSO is the first and foremost algorithm which is the most successful algorithm especially for the solution of the continuous optimization problems. Subsequent algorithms using swarm intelligence nearly have the same ideas. All of them are population based, and

all particles are distributed in the solution space. Particles have initial locations and velocities. They converge to the optimum solution using their swarm intelligence.

Although there have been improvements in the optimization algorithms using swarm intelligence, there is not a unique algorithm which is successful in all types of optimization problems. So the efforts trying to simulate the animal behaviors and swarm intelligence will continue. At the same time, developing hybrid algorithms will also continue until the best combination of the algorithms would be found.

Author details

Pakize Erdogmus

Address all correspondence to: pakizeerdogmus@duzce.edu.tr

Engineering Faculty, Computer Engineering Department, Duzce University, Duzce, Turkey

References

[1] Holland JH. Adaptation in Natural and Artificial Systems. Second edition (First edition, 1975) ed. Cambridge, MA: MIT Press; 1975/1992

[2] Kirkpatrick S, Gelatt C, Vecchi M. Optimization by simulated annealing. Science. 1983; **220**(4598):671-680 Retrieved from http://www.jstor.org/stable/1690046

[3] Kennedy J, Eberhart R. Particle Swarm Optimization. 1995. pp. 1942-1948

[4] Shi Y, Eberhart R. A Modified Particle Swarm Optimizer. In: IEEE International Conference on Evolutionary Computation Proceedings. 1998. pp. 69-73

[5] C. Paper, I. Technology, and T. Kharagpur. Inertia weight strategies in particle swarm inertia weight strategies in particle swarm. no. May 2014, 2011

[6] Clerc M. The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. In: Proceedings of the I999 ICEC. Washington, DC. 1999. pp 1951-1957

[7] Clerc M, Kennedy J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. In: IEEE Transactions on Evolutionary Computation. Feb 2002;**6**(1):58-73. DOI: 10.1109/4235.985692

[8] Eberhart RC, Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512). La Jolla, CA. 2000;**1**:84-88. DOI: 10.1109/CEC.2000.870279

[9] Gao S, Wang H, Wang C, Gu S, Xu H, Ma H. Reactive power optimization of low voltage distribution network based on improved particle swarm optimization. In: Proceedings of the 2017 20th International Conference on Electrical Machines and Systems (ICEMS). Sydney, NSW. 2017. pp. 1-5

[10] Abbas G, Gu J, Farooq U, Asad MU, El-Hawary M. Solution of an economic dispatch problem through particle swarm optimization: A detailed survey - part I. In: IEEE Access. 2017;**5**:15105-15141

[11] Abbas G, Gu J, Farooq U, Raza A, Asad MU, El-Hawary ME. Solution of an economic dispatch problem through particle swarm optimization: A detailed survey – Part II. In: IEEE Access. 2017;**5**:24426-24445

[12] Jois S, Ramesh R, Kulkarni AC. Face localization using skin colour and maximal entropy based particle swarm optimization for facial recognition. In: Proceedings of the 2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON). Mathura, India. 2017. pp. 156-161

[13] Chaudhary R, Patel A, Kumar S, Tomar S. Edge detection using particle swarm optimization technique. In: Proceedings of the 2017 International Conference on Computing, Communication and Automation (ICCCA). Greater Noida, India. 2017. pp. 363-367

[14] Mozaffari MH, Lee WS. Convergent heterogeneous particle swarm optimisation algorithm for multilevel image thresholding segmentation. In: IET Image Processing. 2017; **11**(8):605-619

[15] Karami A, Tafakori L. Image denoising using generalised Cauchy filter. In: IET Image Processing. 2017;**11**(9):767-776

[16] Zhou Xc. Color Image Filter Based on Predator-Prey Particle Swarm Optimization. 2009 International Conference on Artificial Intelligence and Computational Intelligence, Shanghai. 2009. pp. 480-484

[17] Marie-Sainte SL. A new hybrid particle swarm optimization algorithm for real-world university examination timetabling problem. In: Proceedings of the 2017 Computing Conference. London, United Kingdom. 2017. pp. 157-163

[18] Chang JC. Modified particle swarm optimization for solving traveling salesman problem based on a Hadoop MapReduce framework. In: Proceedings of the 2016 International Conference on Applied System Innovation (ICASI). Okinawa. 2016. pp. 1-4

[19] Wenbin G, Yuxin L, Yi W. Energy-efficient job shop scheduling problem using an improved particle swarm algorithm. In: Proceedings of the 2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC). Chongqing. 2017. pp. 830-834

[20] Junaedy A, Sulistijono IA, Hanafi N. Particle swarm optimization for coconut detection in a coconut tree plucking robot. In: Proceedings of the 2017 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC). Surabaya, Indonesia. 2017. pp. 182-187

[21] Roberge V, Tarbouchi M, Labonte G. Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. In: IEEE Transactions on Industrial Informatics. Feb 2013;**9**(1):132-141

[22] Juang C. A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design. 2004;**34**(2):997-1006

[23] Liu B, Wang L, Jin Y, Tang F, Huang D. Improved particle swarm optimization combined with chaos. Chaos, Solitons & Fractals. 2005;**25**:1261-1271

[24] Alatas B, Akin E, Ozer AB. Chaos embedded particle swarm optimization algorithms. Chaos, Solitons & Fractals. 2009;**40**(4):1715-1734

[25] Rong H. An adaptive chaos embedded particle swarm optimization algorithm. In: Proceedings of the 2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering. Changchun. 2010. pp. 314-317

[26] Emrah O, Sinan M, Turhan M. Chaotic quantum behaved particle swarm optimization algorithm for solving nonlinear system of equations. Computers and Mathematics with Applications. 2014

[27] Askarzadeh A. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. Computers & Structures. 2016;**169**:1-12, ISSN 0045-7949. https://doi.org/10.1016/j.compstruc.2016.03.001

[28] Mirjalili S. The ant lion optimizer. Advances in Engineering Software. 2015;**83**:80-98, ISSN 0965-9978. https://doi.org/10.1016/j.advengsoft.2015.01.010

[29] Mirjalili S, Lewis A. The whale optimization algorithm. Advances in Engineering Software. 2016;**95**:51-67. ISSN 0965-9978. https://doi.org/10.1016/j.advengsoft.2016.01.008

[30] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Advances in Engineering Software. 2014;**69**:46-61. ISSN 0965-9978. https://doi.org/10.1016/j.advengsoft.2013.12.007

[31] Wang GG, Deb S, Cui Z. Neural Computing & Applications. 2015. https://doi.org/10.1007/s00521-015-1923-7

[32] Mirjalili S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-Based Systems. 2015;**89**:228-249. ISSN 0950-7051. https://doi.org/10.1016/j.knosys.2015.07.006

[33] Fausto F, Cuevas E, Valdivia A, González A. A global optimization algorithm inspired in the behavior of selfish herds. Biosystems. 2017;**160**:39-55. ISSN 0303-2647. https://doi.org/10.1016/j.biosystems.2017.07.010

[34] Mirjalili S, Amir H, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. Advances in Engineering Software. 2017;**114**:163-191. ISSN 0965-9978. https://doi.org/10.1016/j.advengsoft.2017.07.002

[35] Yang XS. Firefly Algorithms for Multimodal Optimization. In: Watanabe O, Zeugmann T, editors. Stochastic Algorithms: Foundations and Applications. Lecture Notes in Computer Science. Vol 5792. Springer, Berlin, Heidelberg: SAGA; 2009

[36] Passino KM. Biomimicry of bacterial foraging for distributed optimization and control. In: IEEE Control Systems. Jun 2002;**22**(3):52-67. DOI: 10.1109/MCS.2002.1004010

[37] Dorigo M, Maniezzo V, Colorni A. Ant system: Optimization by a colony of cooperating agents. In: IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics). Feb 1996;**26**(1):29-41. DOI: 10.1109/3477.484436

[38] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. J. Global Optimiz. 2007;**39**:459-471

[39] Yang X-S, Deb S. Cuckoo Search via Levy Flights. In: World Congress on Nature & Biologically Inspired Computing (NaBIC 2009). 2009. pp. 210-214

[40] Padmavathi K, Sri Rama Krishna K. Hybrid firefly and Particle Swarm Optimization algorithm for the detection of Bundle Branch Block. International Journal of the Cardiovascular Academy. 2016;**2**(1):44-48. ISSN 2405-8181. https://doi.org/10.1016/j.ijcac.2015.12.001

[41] Dhillon SS, Lather JS, Marwaha S. Multi objective load frequency control using hybrid bacterial foraging and particle swarm optimized PI controller. International Journal of Electrical Power & Energy Systems. 2016;**79**:196-209. ISSN 0142-0615, https://doi.org/10.1016/j.ijepes.2016.01.012

[42] Kora P, Kalva SR. Hybrid bacterial foraging and particle swarm optimization for detecting bundle branch block. SpringerPlus. 2015;**4**(1):481

[43] Vitorino LN, Ribeiro SF, Bastos-Filho CJA. A mechanism based on artificial bee Colony to generate diversity in particle swarm optimization. Neurocomputing. 2015;**148**:39-45. ISSN 0925-2312. https://doi.org/10.1016/j.neucom.2013.03.076

[44] Cheng C-Y, Chen Y-Y, Chen T-L, Yoo JJ-W. Using a hybrid approach based on the particle swarm optimization and ant colony optimization to solve a joint order batching and picker routing problem. International Journal of Production Economics. (Part C). 2015;**170**:805-814. ISSN 0925-5273. https://doi.org/10.1016/j.ijpe.2015.03.021

[45] Li Z, Wang W, Yan Y, Zheng L. PS–ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems. Expert Systems with Applications. 2015;**42**(22):8881-8895. ISSN 0957-4174. https://doi.org/10.1016/j.eswa.2015.07.043

[46] Dash J, Dam B, Swain R. Optimal design of linear phase multi-band stop filters using improved cuckoo search particle swarm optimization. Applied Soft Computing. 2017;**52**:435-445. ISSN 1568-4946. https://doi.org/10.1016/j.asoc.2016.10.024

Chapter 2

Particle Swarm Optimization Algorithm with a Bio-Inspired Aging Model

Eduardo Rangel-Carrillo,
Esteban A. Hernandez-Vargas, Nancy Arana-Daniel,
Carlos Lopez-Franco and Alma Y. Alanis

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.71791

Abstract

A Particle Swarm Optimization with a Bio-inspired Aging Model (BAM-PSO) algorithm is proposed to alleviate the premature convergence problem of other PSO algorithms. Each particle within the swarm is subjected to aging based on the age-related changes observed in immune system cells. The proposed algorithm is tested with several popular and well-established benchmark functions and its performance is compared to other evolutionary algorithms in both low and high dimensional scenarios. Simulation results reveal that at the cost of computational time, the proposed algorithm has the potential to solve the premature convergence problem that affects PSO-based algorithms; showing good results for both low and high dimensional problems. This work suggests that aging mechanisms do have further implications in computational intelligence.

Keywords: particle swarm optimization, bio-inspired aging model, evolutionary optimization algorithms

1. Introduction

Bio-inspired optimization algorithms are based on precise observation of natural systems [1–3]. A relevant characteristic of these algorithms is that the biological process had been tested, validated and proven by means of evolution. The mechanisms of self-adaption, self-organizing and self-learning in natural inspired optimization approaches provide means to address challenging problems that cannot be solved by traditional methods [4].

Thus, bio-inspired algorithms become particularly important to tackle complex optimization problems [6–10]. The outstanding performance of bio-inspired optimization algorithms is attributed

to their structures which are closely related to one or other features observed in nature [10]. Accuracy and repeatability are the prime objectives of every optimization algorithm. Therefore, modeling biological mechanisms may impact the outcome of the observed system, designing more accurate and efficient heuristic algorithms [13, 16].

Problem-solving algorithms inspired by one or more biological features have been developed after observing the behavior in humans, animals, and cells [10–13]. For instance, Genetic Algorithms (GA) [4] defined the basis for evolutionary computing using the early works of Darwin [14] and Mendel [15]; or the Ant Colony System (ACS) [6, 7] which considers the traveling behavior model of self-organized argentine ants published by Goss [5], solving in a fashion way travel salesman-type problems; or the Particle Swarm Optimization (PSO) [8] inspired on feeding behavior of bird flocks becoming a very popular optimizer nowadays [17].

Particle-based optimizers, like those described in [8]; or those presented in [18–22, 28] are very popular because instead of working with one candidate solution, they offer a subset of individual candidate solutions (particles), which are explored, exploited and improved. A relevant mechanism related to evolution that could play a central role in optimization algorithms is aging [23, 25]. Aging is a natural characteristic whose inclusion in a particle-based optimizer could give a mean of individual control over the particle without highly increasing the complexity of algorithms [26].

To the authors' best knowledge, previous PSO algorithms did not have a measurement to control individual particle existence within the swarm by evaluating each particle performance. In PSO, because of the very nature of the algorithm, an effect called *premature convergence* appears when most (or all) of the particles within a swarm compromises their ability to explore and stay close to a local solution. The Particle Swarm Optimization with an Aging Leader and Challengers (ALC-PSO) [24] was the first approach to include aging processes to alleviate this unwanted effect. However, this was only leadership-oriented and not swarm-related; even more important: the aging dynamics were linear and bounded to static predefined values. In [29], it is used to design a high speed symmetric switching CMOS inverter.

Our PSO variant proposal, the Particle Swarm Optimization with Bio-inspired Aging Model (BAM-PSO) is based on a mathematical model that describes the telomeres shortening observed in the immune system cells, this model includes a form of *aging* effect over all the particles of the swarm; this mechanism provides a mean to control the existence of each particle within the swarm avoiding the premature convergence effect. Therefore, the PSO variant with aging model possesses the potential to outperform current optimization algorithms and have further implications in computational intelligence. Finally, optimization under uncertainty and complex functions is an area of special scientific interest, and many real problems and applications include some form of uncertainty; it is also known that collective-intelligence algorithms perform excellent in this type of scenarios [11]; BAMPSO has been successfully implemented in several optimization applications: in time-series forecasting [30] it was implemented as a training algorithm for an artificial neural network and in [31] it was used over a Geometric Algebra (GA) framework in order to compute the rigid movement on images to improve the accuracy of Structure from Motion (SfM) algorithms, which comprises a

family of computer vision algorithms whose paradigm is based on extracting structures when movement is detected or extracting movement when structures are detected in 2D images. Therefore, in this work, BAM-PSO is tested with several popular and well-established benchmark functions and its performance is compared to well-known evolutionary algorithms in both low and high dimensional scenarios.

2. Aging mechanisms to alleviate the premature convergence

Aging is the process of becoming older, which consists on the accumulation of changes over time. This process affects all living systems: humans, cells, unicellular organisms, fruit flies and mammals like rodents [12, 32, 42]. Since the particles of the PSO optimizer algorithm can be treated as a living system, aging could represent a relevant mechanism to alleviate the premature convergence problem in heuristic algorithms. Nowadays, we have better understanding of the lifespan of human cells, which is determined by homoeostatic properties of the immune system. Homeostasis refers to the regulation of the lymphocytes pool in an organism. It is assumed that the number of cells is determined by the capacity of the peripheral immune system.

In the immune system, it is observed that cell death rate accelerates if the immune cells exceed the allocated free space [33]. For instance, in the course of a viral infection, immune system cells can undergo approximately 15–20 divisions. Total proliferative capacity of human T lymphocyte is about 40–45 divisions and depends on the telomere length [41]. Telomeres are the end parts of the chromosomes, which become shorter in every cell division; this can be appreciated in **Figure 1**. The cell can reach its unresponsiveness state when the telomere length completes about one half of its initial value.

Telomere dynamics can be interpreted in a mathematical model based on experimental observations. In this work, we consider the mathematical model proposed by [33] to represent the telomere dynamics. This model considers the following equation:

$$\frac{dT}{dt} = \alpha(p^* - T)N \tag{1}$$

where T represents the remaining telomere divisions per cell. α defines the telomere consumption rate per iteration. p^* represents the length of telomere repeats in naive cells produced at the age t (with initial length $p_0 = 8.3 \times 10^3$) and N is the number of cells as defined in [33].

Eq. (1) is a differential non-ascending equation that defines the derivative in telomere division per cell depending on the consumption rate of the cell α, telomere capacity of the cell (p^*) and number of cells (N). Eq. (1) describes the dynamic of average telomere length T in the pool of naive cells. The rate of this process depends on $(p^* - T)$, where p^* is the telomere length in the cells and p_0 defines the telomere at initial age. This dynamic describes the self-sustaining process of regulation of total concentration of the T cells and how the telomere is affected by T cells concentration and iterations [33].

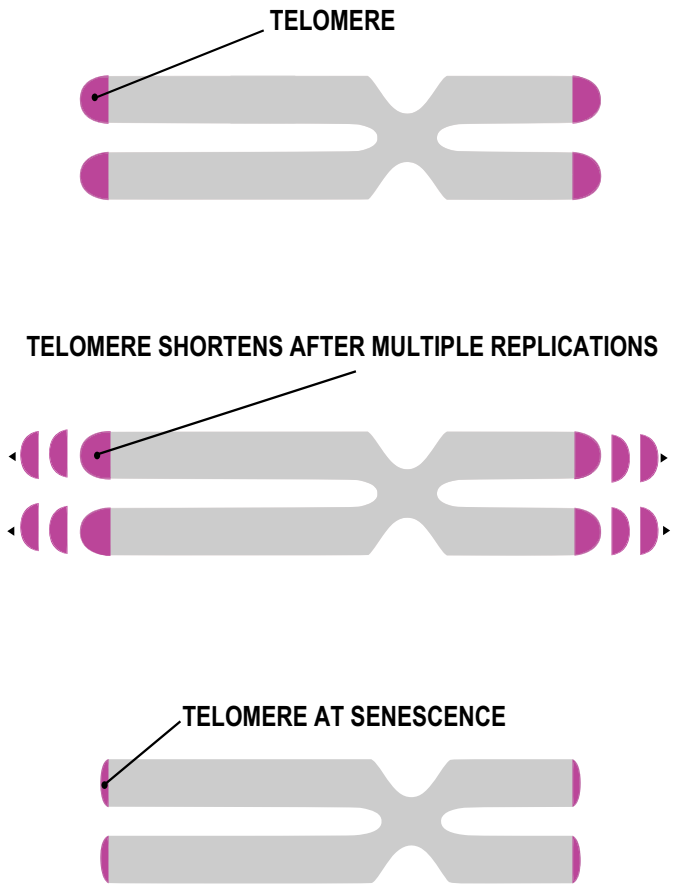

Figure 1. Telomere division process in human *T* cells.

The proposed scheme in Eq. (1) can provide PSO the same self-regulation capabilities during swarm's concentration around a local minimum (known as premature convergence) if we consider the swarm as lymphocytes, and particle concentration around a local minimum as *T* cells concentration. Finally, senescence of the particle will be translated as a new random-generated particle, replacing the unresponsive one. For this to be achieved, a given particle within the swarm will have a limited number of iterations to exist within the swarm, similar to telomere length at p_0, and senescence of this particle will occur when particle's capacity for search space-exploitation approaches to 0 and swarm's concentration around a local minimum has exceeded a given limit (premature convergence indicator), similar to *T* cells in human immune systems [35].

Based on this aging mechanism, it is possible to include senescence to a given particle within the swarm, and the lifespan of each particle will be adjusted according to the error produced by its candidate solution and the premature convergence indicator of the swarm.

3. Particle swarm optimization with a bio-inspired aging model (BAM-PSO)

The ALC-PSO variant [24] suggests an interesting approach using aging factors to define when to remove non-useful characteristics of the algorithm. However, this variant proposes a simple aging model with the following characteristics:

- The lifespan exists only for the *leader* and is controlled by linear means according to lifespan controller shown in [24].
- There is no lifespan controller for the rest of the particles of the swarm, meaning that the particles continue offering candidate solutions even if premature convergence has occurred.
- There is no communication between particles within the swarm about premature convergence and no evaluation is performed about particle concentration around local minima.

Consequently, there are several points that can be improved in ALC-PSO. For instance, including aging mechanisms to the rest of the particles within the swarm may help exploration without affecting the convergence. Moreover, in order to alleviate premature convergence in the PSO, there is an urgent need to include means of measuring the premature convergence in real time allowing the swarm to discard non-useful particles and to explore new candidate solutions without losing the convergence inertia toward the global minimum.

Based on previous observations, we propose a variant of the PSO named Particle Swarm Optimization with a Bio-inspired Aging Model (BAM-PSO). Our proposed algorithm considers the aging leader and challengers in the same fashion as ALC-PSO, but it applies senescence to each particle within the swarm by using the mathematical model that describes aging dynamics in Eq. (1).

For BAM-PSO to implement senescence efficiently, it is necessary to implement a mechanism that allows the algorithm to interpret when the swarm has reached a local minimum; this can be achieved by means of premature convergence measurement.

In the aging model represented by Eq. (1), the number of cells can be interpreted as a measure of the particles numbers around the same one-dimensional location. In this sense, measuring the standard deviation among the swarm in each particle dimension can be computed as follows:

$$k_j = \frac{k_{min}}{\sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{D}\left(x_{ij}-\overline{x_j}\right)^2}{(D-1)}}} \tag{2}$$

where $D \in \mathbb{R}$ is the dimension of the problem; k_{min} represents the deviation minimum for all dimensions; k_j is the premature convergence around *j-th* element of the dimension D. Note that x_{ij} is the particle within the swarm and $\overline{x_j}$ represents the mean value of all *j-th* elements of the

swarm. This scheme provides a premature convergence measurement mechanism around each element of the problem dimension.

3.1. Lifespan controller

The BAM-PSO algorithm considers Eq. (2) to evaluate swarm's efficiency, and to control the lifespan of each particle, the aging mechanism proposed by [33] is adapted to the algorithm, satisfying the next criteria:

$$L_{ij} = L_{ij} - \alpha k_j \tag{3}$$

$$\text{With}: \ 0 \leq L_{ij} \leq L_{max}$$

where L_{ij} is the lifespan of the *i-th* particle with j-*th* element of dimension D. L_{max} represents the maximum lifespan of any particle within the swarm.

The error improvement of the particle with respect to the iteration t is calculated by:

$$\alpha = \frac{e_i(t)}{e_i(t-1)} \tag{4}$$

The error of the *i-th* particle $e_i(t)$ is computed within the swarm at iteration t.

This scheme completes the bio-inspired, population-broad aging mechanism and will allow us to propose the final algorithm.

The steps involved in the BAM-PSO algorithm are as follows:

Step 1: Initialization. The initial positions of all particles are generated randomly within the n-dimensional search space, with velocities initialized to *0*. The best particle among the swarm is selected as the *leader.* The age of the *leader* and all particles within the swarm is initialized to *0*.

Step 2: Velocity and position updating. Every particle follows the velocity update rule and the position update rule presented in [8]:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 R_1 \left(p_{ij}(t) - x_{ij}(t)\right) + c_2 R_2 \left(p_{gj}(t) - x_{ij}(t)\right) \tag{5}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{6}$$

with:

$$i = 1, 2, \ldots, S \qquad j = 1, 2, \ldots, D$$

where i is the *ith* particle of a swarm that satisfies $S \in \mathbb{R}^D$, and j is the *jth* element of dimension problem D. Also t represents the iteration counter, R_1 and R_2 are random, normalized and uniformly distributed values. c_1, c_2 represents the social and cognitive parameter, $x_{ij}(t)$ is the particle *ij* position for t iteration, $x_{ij}(t+1)$ is the particle *ij* position for $t+1$ iteration, $v_{ij}(t)$ is the particle's *ij* velocity for t iteration. $p_{ij}(t)$ represents the local best position for particle *ij* in iteration t and $p_{gj}(t)$ represents the global best position for entire swarm in iteration t.

Step 3: Evaluate the *leader* or generate new challengers for leadership according to leadership term according to lifespan controller defined in [24]:

$$\delta_g Best = f\left(p_{gj}(t)\right) - f\left(p_{gj}(t-1)\right) \leq 0 \tag{7}$$

$$\delta_l Best = \sum_{i=1}^{M} f\left(p_{ij}(t)\right) - \sum_{i=1}^{M} f\left(p_{ij}(t-1)\right) \leq 0 \tag{8}$$

$$\delta_{Leader} = f(leader(t)) - f(leader\ (t-1)) \leq 0 \tag{9}$$

with leadership term: $\theta = 1, 2, \ldots, \Theta$.

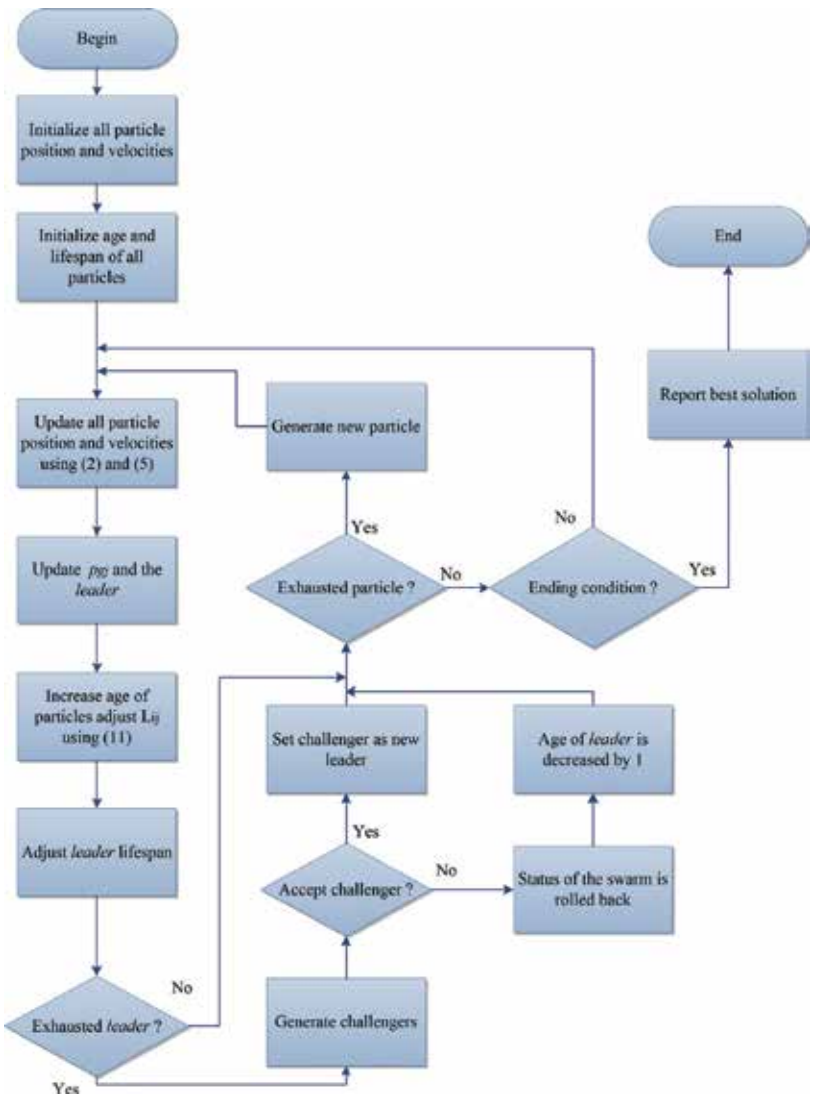

Figure 2. BAM-PSO flow diagram.

where $f(^*)$ represents the objective function value for the best candidate solution, θ the remaining leadership's term, Θ represents the maximum leadership term, δ_g *Best* defines the entire swarm improvement factor, δ_l *Best* represents the individual particle improvement factor, *leader* represents the particle within the swarm that is the acting leader (not necessarily $p_{gj}(t)$) and whose all particles will follow according to Eqs. (5) and (6); finally, δ_{Leader} represents the *leader's* individual improvement factor.

Eqs. (7), (8) and (9) indicate the leading performance of the *leader.* The lifespan controller utilizes these performance evaluations to adjust the leading term of *leader* according to the following decision tree:

if $\delta_g Best < 0$: $\theta = \theta + 2$ $(up\ to\ \Theta)$, else :

if $\delta_l Best < 0$: $\theta = \theta + 1$ $(up\ to\ \Theta)$.else :

if $\delta_{Leader}\theta < 0$: $\theta = \theta$ $(no\ increase)$, else :

$$\theta = \theta - 1\ (leader\ term's\ reduction).$$

When the leading term of *leader* reaches $\theta = 0$ the *leader* is considered exhausted and replaced by newly generated challengers as described in [24].

Step 4: Adjust lifespan of all particles within the swarm according to Eqs. (2)–(4) and replace particles with random ones for every depleted lifespan.

Step 5: Terminal condition check. If the number of iterations is larger than the predefined or the error has reached a minimum expected value, the algorithm terminates. Otherwise go to Step 2 for a new round of iteration.

Figure 2 shows the flow chart for BAM-PSO algorithm.

4. Results

The proposed BAM-PSO algorithm is compared with five different biologically inspired algorithms: PSO with inertial vector and boundaries [27], ant colony system (ACS) [6], differential evolution (DE) [31], simplified swarm optimization (SSO) [20] and particle swarm optimization with aging leader and challengers (ALC-PSO) [24]. These algorithms are selected because of several factors: first, PSO is the base algorithm for BAM-PSO, so it is natural to compare performance with the original optimizer, SSO and ALC-PSO are other well-known variants of PSO that in some way, claim to alleviate the premature convergence problem and, specifically, ALC-PSO is related in many ways to BAM-PSO. Finally, while ACS and DE are not related closely to BAM-PSO, they are swarm-based and evolution-based optimization algorithms respectively and thus, were considered as good candidates for performance comparison.

To test optimization performance of these algorithms, well-established benchmark functions are selected in low and high dimensionality [33]. These selected functions help evaluate algorithm's performance over a broad type of problems, because they possess multiple local

minima, complex non-linear structure, or have bowl-shaped/plate-shaped structure [36, 37]; even some of them have a steep ridge and drops structure. From the literature, a list of 18 functions was considered relevant enough to test BAM-PSO performance. The selected benchmark functions are shown in **Table 1**.

For comparison purposes, all algorithms were configured in similar vein when it was possible, e.g. the ACS algorithm [6] uses *ant-type* vectors which can be considered particles in a *swarm* like those found in the PSO [8, 21], ALC-PSO [24] and the proposed BAM-PSO algorithms. Nevertheless, the behavior and setting are very different, since *ant-type* vectors behavior is determined by a mathematical model that simulates the pheromone attraction between biological ants. The DE algorithm [34] does not have a *swarm*-based mathematical model for the dynamics of particles, but instead the mathematical model used to simulate *evolution* is based on vectors and mutation factors. Finally, SSO algorithm [20] does not consider linear equations

f_1	Griewank	$1+\frac{1}{4000}\ \sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$
f_2	Ackley	$-20e^{\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)}-e^{\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right)}+20+e$
f_3	Sphere	$\sum_{i=1}^{n} x_i^2$
f_4	Rastrigin	$A_n+\sum_{i=1}^{n}\left[x_i^2+A\cos 2\pi x_i\right]$
f_5	Zakharov	$\sum_{i=1}^{n} x_i^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^4$
f_6	Trid	$\sum_{i=1}^{n}(x_i-1)^2-\sum_{i=1}^{n}x_ix_{i-1}$
f_7	Sum of squares	$\sum_{i=1}^{n} ix_i^2$
f_8	Sum of exponential squares	$\sum_{i=1}^{n}\left[\left(\sum_{j=1}^{n}x_j^i\right)-b_i\right]^2$
f_9	Styblinski-Tang	$\frac{1}{2}\sum_{i=1}^{n}\left(x_i^4-16x_i^2+5x_i\right)$
f_{10}	Shifted Rastrigin	$\sum_{i=1}^{n}\left[A+\sum_{i=1}^{n}\left[x_i^2-A\cos 2\pi x_i\right]\right.$
f_{11}	Schwefel 1.2	$\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$
f_{12}	Elliptical	$\sum_{i=1}^{n}10^{6\frac{i-1}{d-1}}x_i^2$
f_{13}	Rosenbrock	$\sum_{i=1}^{n}\left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$
f_{14}	Schwefel	$418.9829d-\sum_{i=1}^{n}x_i sen\left(\sqrt{\lvert x_i\rvert}\right)$
f_{15}	Perm D, 0, β-function	$\sum_{i=1}^{n}\left(\sum_{j=1}^{n}\left(j^i+\beta\right)\left(\left(\frac{x_i}{j}\right)^i-1\right)\right)^2$
f_{16}	Michalewicz	$-\sum_{i=1}^{n} sen(x_i)sen^{2m}\left(\frac{ix_i}{\pi}\right)$
f_{17}	Levy	$sen^2(\pi\omega_1)+\sum_{i=1}^{n-1}(\omega_i-1)^2\left[1+10sen^2(\pi\omega_i+1)\right.$ $\left.+(\omega_n-1)^2[1+sen2\pi x_n]\right];\ \omega=1+\frac{x_i-1}{4}$
f_{18}	Dixon-Price	$(x_i-1)^2+\sum_{i=2}^{n} i\left(2x_i^2-x_i-1\right)^2$

Table 1. Benchmark functions used in algorithm performance comparison for BAM-PSO.

to update the information of the particles, instead a probability function is considered to decide the next particle position based on previously defined settings.

4.1. Evaluating the algorithms in low dimensional settings

The swarm size S for every algorithm is set to *20*, dimension D for every function is set to 2, and total iterations are set to *10,000* for each objective function. **Table 2** reveals the performance for the different selected algorithms in a low dimension, the results show the best possible solution offered by the algorithm after terminal condition was reached. As we can see, both BAM-PSO and ALC-PSO algorithms show improved performance in comparison to the other algorithms. Meaning that BAM-PSO provides good results in low dimensional problems for all the benchmark functions, outperforming most of the other tested algorithms. It is important to note, that results marked in Bold are the best solution obtained for each case.

4.2. Evaluating the algorithms in high dimensional settings

Our second simulation scenario consists in evaluating the performance of the BAM-PSO with high dimensional problems. In this case, the total of 18 benchmark functions from **Table 1** was considered and the function dimension D was configured to *30*.

Based on the previous results, ALC-PSO, SSO, and PSO algorithm were selected to compare results with the BAM-PSO because of their shared origin. However, ACS was also included due to its swarm nature.

At first glance, the results shown in **Table 3** suggest that the BAM-PSO provides the best performance of all compared algorithms in highly-dimensional problems for several benchmark functions. It is important to note, that results marked in Bold are the best solution obtained for each case.

4.3. Non-parametrical statistical analysis of BAM-PSO performance results

In order to conclude whether or not BAM-PSO outperforms the other selected algorithms, more accurate means of comparison other than simple observation of benchmark results are required; for this reason, some of the most popular non-parametric statistical tests were employed. This type of analysis is widely accepted as a metric of performance comparison between algorithms in a pair-wise configuration [43]. To this end, using the statistical procedures defined by [38–40], the Signed Test and the Wilcoxon Test statistical analysis were selected.

Dimension = 2	BAM-PSO	ALC-PSO	DE	SSO	ACS	PSO
f_1	**0.000000E + 00**	**0.000000E + 00**	7.800000E-03	4.250000E-02	4.880000E-09	**0.000000E + 00**
f_2	**8.880000E-16**	**8.880000E-16**	2.120000E + 00	4.300000E-02	4.160000E-02	**8.880000E-16**
f_3	**0.000000E + 00**	**0.000000E + 00**	2.800000E-01	5.460000E-06	5.460000E-08	3.550000E-43
f_4	**0.000000E + 00**	**0.000000E + 00**	2.610000E + 00	4.300000E-03	7.300000E-01	**0.000000E + 00**

Table 2. Optimization results comparison for D = 2.

Dimension = 30	BAM-PSO	ALC_PSO	PSO	SSO	ACS
f_1	3.420000E-01	4.810000E-02	**1.664995E-02**	1.017972E + 00	1.009513E + 00
f_2	1.710000E + 00	2.950000E + 00	**6.179332E-01**	1.936833E + 01	1.575243E + 01
f_3	9.720000E-04	1.060000E + 00	**1.054778E-05**	3.628362E + 02	2.766939E + 02
f_4	**3.830000E + 00**	1.420000E + 03	2.003442E + 02	1.421872E + 06	4.577661E + 05
f_5	**2.701825E-07**	1.943493E-01	1.314056E + 01	6.189616E + 02	5.320685E + 05
f_6	**1.000012E + 00**	1.347625E + 04	5.670018E + 03	2.350724E + 06	3.796814E + 06
f_7	**1.838826E-25**	1.846162E-01	8.309639E-01	5.267949E + 03	7.131057E + 03
f_8	**1.693373E-15**	5.924127E-08	2.569830E-10	5.672124E-02	2.042905E-01
f_9	1.591141E + 00	**4.989256E-03**	1.084663E + 03	6.528900E + 02	4.572974E + 02
f_{10}	**4.265668E-03**	2.575325E-01	6.748519E + 01	2.362575E + 01	6.768979E + 01
f_{11}	1.000007E + 00	**0.000000E + 00**	8.449218E + 120	3.61590E + 115	1.11140E + 121
f_{12}	**1.00008E + 119**	9.432255E + 119	2.858556E + 125	1.96354E + 124	3.06278E + 125
f_{13}	**1.000053E + 00**	9.837181E + 00	5.033260E + 01	3.015314E + 03	3.832691E + 03
f_{14}	**1.005478E + 00**	2.327770E + 03	9.314781E + 03	8.149385E + 03	1.062061E + 04
f_{15}	2.039775E-23	**0.000000E + 00**	8.520128E-06	4.815499E-04	9.508960E + 02
f_{16}	**9.177340E + 00**	1.276359E + 01	1.683612E + 02	1.086427E + 02	4.458898E + 01
f_{17}	**1.000155E + 00**	6.796335E + 00	1.001832E + 00	1.026017E + 02	1.676198E + 02
f_{18}	**9.112266E-01**	2.056725E + 02	4.960351E + 00	7.179904E + 05	8.280520E + 05

Table 3. Optimization results comparison for D = 30.

In **Table 4**, we can observe that BAM-PSO outperforms the other algorithms with an accepted level of significance using this procedure. However, this test is a simple first-line procedure and to uncover more evidence over the results, we rely on a more robust and sensitive procedure, which is the Wilcoxon Test.

The Wilcoxon test results shown in **Table 5** shed light over the fact that BAM-PSO algorithm can go beyond the results provided by the PSO, SSO and ACS with great statistical significance ($P < 0.001$), but the procedure finds not enough evidence to conclude that the BAM-PSO can outperform the ALC-PSO at this level of statistical significance; however, the results are good enough to show that BAM-PSO can outperform ALC-PSO with good statistical significance ($P < 0.01$), and considering the literature claim that: if the resulting P-value is small enough ($P < 0.05$), then it can be accepted that the median of the differences between the paired

BAM-PSO vs.	ALC_PSO	PSO	SSO	ACS
Positive results	14	15	18	18
Negative results	4	3	0	0
Significant difference? (P < 0.05)	**Yes**	**Yes**	**Yes**	**Yes**

Table 4. Non-parametrical sign test for benchmark results at D = 30.

BAM-PSO	(R+) positive ranks obtained	(R−) negative ranks obtained	(Rm) maximum negative ranks	Accepted significance? ($P < 0.05$)
vs ALC-PSO	147.0	24.0	27 for ($P < 0.01$)	**Yes**
vs PSO	156.0	15.0	17 for ($P < 0.001$)	**Yes**
vs SSO	171.0	0.0	17 for ($P < 0.001$)	**Yes**
vs ACS	171.0	0.0	17 for ($P < 0.001$)	**Yes**

Table 5. Non-parametrical Wilcoxon test for benchmark results at D = 30.

observations is statistically significantly different from 0 [44]. We can conclude then, that BAM-PSO has a greater performance over a broad set of benchmark functions over all other selected algorithms with statistical relevance, including ALC-PSO.

The performance of BAM-PSO can be explained by its senescence mechanism: after particles falls into local minimum, they offer less improvement; then, the senescence mechanism starts acting by producing *senescence* on the swarm; then, exhausted particles are replaced with random ones through the search space. This favors exploration after premature convergence without completely eliminating exploitation of search space near the local minimum, which in the end provides better optimization results than other PSO variants.

5. Conclusions

In this chapter, we introduced a PSO variant algorithm called Particle Swarm Optimization with Bio-inspired Aging Model (BAM-PSO) which was compared with other five popular bio-inspired optimizers. This test was performed using popular benchmark functions with low and high dimensionality configuration.

We observed that the BAM-PSO algorithm has the potential to solve the premature convergence problem of PSO showing good results for both low and high dimensional problems with statistical relevance according to several non-parametric analyses. Furthermore, according to results shown in Section 4, BAM-PSO performs better than the selected PSO variants.

As shown in results section, BAM-PSO outperforms all other compared swarm-based algorithms with at least a confidence factor as high as $P < 0.01$. However, the cost of this improved accuracy is found in computation complexity due to the introduction of Eq. (2) and all the lifespan control for the particles; which in turn translates to computing time; this time increase was found to be approximately of at least 9 times the required computation time for the original PSO on the conducted experiments of section 5 and 1.5 times the required computation time for ALC-PSO. However, this increase in time is not fixed, as it depends on how early the premature convergence occurs and how many particles are replaced after senescence.

Finally, these experimental results provide support on the important role of aging mechanisms during the selection process in bio-inspired optimization algorithms, because the population-broad

aging mechanism implemented in BAM-PSO allows the algorithm to provide better results than some other popular optimizers that does not implement aging.

Acknowledgements

The authors thank the support of CONACYT Mexico, through Projects CB256769 and 340 CB258068 (Project supported by Fondo Sectorial de Investigación para la Educación).

Author details

Eduardo Rangel-Carrillo[1], Esteban A. Hernandez-Vargas[2], Nancy Arana-Daniel[1], Carlos Lopez-Franco[1] and Alma Y. Alanis[1]*

*Address all correspondence to: almayalanis@gmail.com

1 CUCEI, Universidad de Guadalajara, Guadalajara, Jalisco, Mexico

2 Frankfurt Institute for Advanced Studies, Frankfurt am Main, Germany

References

[1] Von Neumann J. Zur theorie der gesellschaftsspiele. Mathematische Annalen. 1928;**100** (1):295-320. DOI: 10.1007/BF01448847

[2] Maynard J, Price R. The logic of animal conflict. Nature. 1973;**5427**(246):15-18. DOI: 10.1038/246015a0

[3] Henson S, Hayward J. The mathematics of animal behavior: An interdisciplinary dialogue. Notices of the AMS. 2010;**57**(10):1248-1258

[4] Barricelli N. Esempi numerici di processi di evoluzione. Methodos. 1954;**6**(21–22):45-68

[5] Goss S, Aaron S, Deneubourg J, Pasteels J. Self-organized shortcuts in the angentine ant. Naturwissenschaften. 1989;**76**(12):579-581. DOI: 10.1007/BF00462870

[6] Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies. Proceedings of ECAL91. 1991;**1**(1):134-142. DOI: 10.1007/BF00462870

[7] Socha K, Dorigo M. Ant colony optimization for continuous domains. European Journal of Operational Research. 2008;**1**(1):1155-1173. DOI: 10.1016/j.ejor.2006.06.046

[8] Kennedy J, Eberhart R. Particle swarm optimization. Proceedings of the IEEE International Joint Conference on Neural Networks. 1995;**4**(1):1942-1948. DOI: 10.1109/ICNN.1995.488968

[9] Vrahatis M. Particle Swarm Optimization and Intelligence, Advances and Applications. 1st ed. Hershey, PA, USA: Information Science Publishing (IGI Global); 2010. p. 328. DOI: 10.13140/2.1.3681.1206

[10] Castro L. Fundamentals of natural computing: An overview. Physics of Life Reviews. 2007;**4**(1):1-36. DOI: 10.1016/j.plrev.2006.10.002

[11] Simon D. Evolutionary Optimization Algorithms. 1st ed. 1. New Jersey, USA: John Wiley & Sons; 2013. 772 p. DOI: 978-0-470-93741-9

[12] Gershon H, Gershon D. The budding yeast, saccharomyces cerevisiae, as a model for aging research: A critical review. Mechanisms of Ageing and Development. 2000;**120**(1–3): 1-22. DOI: 10.1016/S0047-6374(00)00182-2

[13] Fisher R. The Genetical Theory of Natural Selection. 1st ed. New York, USA: Nabu Press; 1930. p. 360. DOI: 0-19-850440-3/ISBN 9780198504405

[14] Darwin C. The Origin of Species. 1st ed. United Kingdom of Great Britain and Ireland: Fellow of the Royal, Geological, Linnaean, Etc., Societies; 1859. p. 502. DOI: 10.5962/bhl.title.46292

[15] Mendel G. Verhandlungen des naturforschenden ver-eines in Brunn. Versuche Uber Plflanzen-hybriden. 1865;**IV**(1):3-47. DOI: 10.5962/bhl.title.61004

[16] Grosan C, Abraham A. Hybrid evolutionary algorithms: Methodologies, architectures, and reviews. Studies in Computational Intelligence. 2007;**75**(1):1-17. DOI: 10.1007/978-3-540-73297-6-1

[17] Van Den Bergh F, Engelbrecht A. A cooperative approach to particle swarm optimization. IEEE Transactions on Evolutionary Computation. 2004;**8**(3):225-239. DOI: 10.1109/TEVC.2004.826069

[18] Meissner M, Schmuker M, Schneider G. Optimized particle swarm optimization (OPSO) and its application to artificial neural network training. BMC Bioinformatics. 2006;**7**(1): 1-11. DOI: 10.1186/1471-2105-7-125

[19] Jiang B, Wang N, Li X. Particle swarm optimizer with aging operator for multimodal function optimization. International Journal of Computational Intelligence. 2014;**6**(5):862-880. DOI: 10.1080/18756891.2013.807108

[20] Changseok B, Wei-Chang Y, Noorhanizan W, Yuk-Ying C, Liu Y. A new simplified swarm optimization. International Journal Of Innovative Computing Information And Control. 2012;**8**(6):4391-4406. DOI: 1349-4198

[21] Chen G, Huang K. Online parameter identification of an induction motor using improved particle swarm optimization. In: IEEE, editor. IEEE Control Conference; 26–31-07-2007; Hunan, China. Hunan, China: IEEE; 2007. pp. 745-749. DOI: 10.1109/CHICC.2006.4347151

[22] Liu C, Du W, Wang W. Particle swarm optimization with scale-free interactions. PLoS One. 2014;**9**(5):1-8. DOI: 10.1109/4235.985692

[23] Rahman H. Ainul Hayat by Muhammad Ibn Yusuf Al-Harawi. 1st ed. India: Ibn Sina Academy of Medieval Medicine and Sciences; 2007

[24] Wei-Neng C, Jun Z, Ying L, Chen N, Zhi-Hui Z, Henry S, et al. Particle swarm optimization with aging leader and challengers. IEEE Transactions on Evolutionary. 2013;**17**(2): 241-258

[25] Williams G. Pleiotropy, natural selection, and the evolution of senescence. Evolution. 1957;**11**(4):398-411. DOI: 10.2307/2406060

[26] Clerc M, Kennedy J. The particle swarm – explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation. 2002;**6**(1):58-73. DOI: 10.1109/4235.985692

[27] Eberhart R, Shi Y. Comparison between genetic algorithms and swarm optimization. Evolutionary programming VII. 1998;**1447**(1):611-616. DOI: 10.1007/BFb0040812

[28] Van den Bergh F, Engelbrecht A. A new locally convergent particle swarm optimiser. In: IEEE, editor. IEEE International Conference on Systems, Man and Cybernetics; 6–9 Oct 2002; Yasmine Hammamet, Tunisia, Tunisia. Yasmine Hammamet, Tunisia, Tunisia: IEEE; 2002. DOI: 10.1109/ICSMC.2002.1176018

[29] Prasad B, Mandal D, Ghoshal SP. PSO with aging leader and challengers for optimal design of high speed symmetric switching CMOS inverter. International Journal of Machine Learning and Cybernetics. 2016;**8**(4):1403-1422. DOI: 10.1007/s13042-016-0517-z

[30] Rangel E, Alanis AY, Ricalde LJ, Arana-Daniel N, Lopez-Franco C. Bio-inspired aging model particle swarm optimization neural network training for solar radiation forecasting. Progress in pattern recognition, image analysis, computer vision, and applications. CIARP 2014. Lecture Notes in Computer Science. 2014;**8827**(1):682-689. DOI: 10.1007/978-3-319-12568-8_83

[31] Arana-Daniel N, Villaseñor C, Lopez-Franco C, Alanis AY. Bio-inspired aging model-particle swarm optimization and geometric algebra for structure from motion. Progress in pattern recognition, image analysis, computer vision, and applications. CIARP 2014. Lecture Notes in Computer Science. 2014;**8827**(1):762-769. DOI: 10.1007/978-3-319-12568-8_83

[32] Partridge L, Gems D. Benchmarks for aging studies. Nature. 2007;**450**(7167):165-167. DOI: 10.1038/450165a

[33] Romanyukha A, Anatoli I. Age related changes in population of peripheral T cells: Towards a model of immunesenescence. Mechanisms of Ageing and Development. 2013;**124**(1):433-443. DOI: 10.1016/S0047-6374(03)00019-8

[34] Storn R. Kenneth. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization. 1997;**11**(4):341-359. DOI: 10.1023/A:1008202821328

[35] Cibelli J, Blackwell C, Cristofalo V, Francis M, Baerlocher G, al e. Extension of cell life-span and telomere length in animals cloned from senescent somatic cells. Science. 2000;**288**(1):665-669. DOI: 10.1126/science.288.5466.665

[36] Liang J, Qu B, Suganthan P, Hernández-Díaz A. Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization. Technical Report 2012 on Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University. 2013;**1**(1)

[37] Wei-Hong L, Ashidi N, Isa M. Adaptive division of labor particle swarm optimization. Expert Systems with Applications: An International Journal. 2015;**42**(14):5887-5903. DOI: 10.1016/j.eswa.2015.03.025

[38] Ghosh S, Das S, Kundu D, Suresh K, Abraha A. Inter-particle communication and search-dynamics of lbest particle swarm optimizers: An analysis. Informative Science. 2012;**182** (1):156-168. DOI: 10.1016/j.ins.2010.10.015

[39] Derrac J, García S, Molina D, Herrera F. A practical tutorial on the use of nonparametric statistical tests as a methodology. Swarm and Evolutionary Computation. 2011;**1**(1):3-18. DOI: 10.1016/j.swevo.2011.02.002

[40] Blaise L. Technische Universität Carolo-Wilhelmina zur Braunschweig. Master thesis [thesis]. Germany; 2015

[41] Poblete-Castro I, Binger D, Rodrigues A, Becker J, Martins Dos Santos V, Wittmann C. In-silico-driven metabolic engineering of pseudomonas putida for enhanced production of polyhydroxyalkanoates. Metabolic Engineering. 2013;**15**(1):113-123. DOI: 10.1016/j.ymben.2012.10.004

[42] Hernandez-Vargas E, Esther W, Canini L, Toapanta F, Binder S, Uvarovskii A, al e. Effects of aging on influenza virus infection dynamics. Journal of Virology. 2014;**88**(8):4123-4154. DOI: 10.1128/JVI.03644-13

[43] Johnson D. The insignificance of statistical significance testing. Journal of Wildlife Management. 1999;**63**(3):763-772. DOI: 10.2307/3802789

[44] Conover W. Practical Nonparametric Statistics. 3rd ed. New Jersey, USA: Wiley & Sons; 1999. p. 584. DOI: 978-0471160687

Chapter 3

Particle Swarm Optimization Solution for Power System Operation Problems

Mostafa Kheshti and Lei Ding

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.72409

Abstract

Application of particle swarm optimization (PSO) algorithm on power system operation is studied in this chapter. Relay protection coordination in distribution networks and economic dispatch of generators in the grid are defined as two of power system-related optimization problems where they are solved using PSO. Two case study systems are conducted. The first case study system investigates applicability of PSO on providing proper overcurrent relay settings in the grid, while in the second case study system, the economic dispatch of a 15-unit system is solved where PSO successfully provides the optimum power output of generators with minimum fuel costs to satisfy the load demands and operation constraints. The simulation results in comparison with other methods show the effectiveness of PSO against other algorithms with higher quality of solution and less fuel costs on the same test system.

Keywords: power system, economic dispatch, relay coordination, particle swarm optimization

1. Introduction

Electric power system is the most complex man-made system, and the modern society depends heavily on continuous and reliable operation of this system to supply electricity to commercial, residential and industrial consumers. Operation of the power grid involves a balanced platform in generation, transmission and distribution, which costs billions of dollars to run. The reliable and continuous availability of electricity with minimum costs is the major objective of utility grids and energy providers. Two of the important complex problems in power system are economic dispatch and power system protection.

The power plants and utility grids need to allocate the available generation units in an efficient and economical way to respond to the load demand in order to provide continuous power supply in stable conditions and with minimum power production costs. This is addressed as

economic dispatch (ED) [1, 2]. With the practical constraints on the generators, finding optimum power outputs with minimum fuel costs is challenging.

In addition, as the occurrence of failures and faults in the power grid is inevitable, the entire power system must be protected. The relay protection scheme is designed to detect faults and isolate the faulty parts of the grid from the healthy sections in order to mitigate the consequences of the faults and maintain continuity of service. If a fault occurs, the nearest corresponding relays must operate as fast as possible to clear the fault. If due to any reason these primary relays fail to react, their backup relays must operate and accomplish the task. Directional overcurrent relays (DOCRs) are a suitable and economical protection scheme for distribution systems [3]. The protection design of DOCRs is based on two parameters, time multiplier setting (TMS) and plug setting (PS). Proper settings of TMS and PS allow a primary relay to clear the faults in its protection zone as fast as possible and in case of failure, its backup relay operates immediately after a time interval to clear the fault. TMS and PS values of each relay must be coordinated with other backup relays, where again relays act with different current settings, which make the coordination a complex task. Each pairs of relays include four variables (TMS, PS) and the complexity of coordination will be intense in bigger systems with more relays and constraints. Due to the complex interconnection of the distribution systems and also nonlinear characteristics of operation time of relays, finding best relay settings could be very difficult.

Considering the non-convex and nonlinear nature of these problems, traditional methods fail to feasibly or optimally solve them. Therefore, evolutionary algorithms have gained more attentions as solutions to such optimization problems. Some of the recent related works on ED problem have been studied with the metaheuristic methods such as Genetic Algorithm (GA) [4], Particle Swarm Optimization (PSO) [4], Imperialist Competitive Algorithm (ICA) [5], Artificial Bee Colony (ABC) [6], Bacterial Foraging Optimization (BF) [7], Hybrid Harmony Search with Arithmetic Crossover (ACHS) [8], GA with a special class of ant colony optimization (GAAPI) [9] and so on. The modified and hybrid models of PSO such as Modified PSO (MPSO) [10], guaranteed convergence PSO (GCPSO) [10], Species-based Quantum Particle Swarm Optimization (SQPSO) [11], Iteration PSO (IPSO) [12], Parallel PSO with Modified Stochastic Acceleration Factors (PSO-MSAF) [13], Distributed Sobol PSO and Tabu Search Algorithm (DSPSO-TSA) [14], Self-Organizing Hierarchical PSO (SOH-PSO) [15], Passive Congregation-based PSO (PC-PSO) [15] and Simple PSO (SPSO) [15] have also been employed to address the ED problem.

Application of metaheuristic algorithms on power system protection and particularly, DOCR coordination in distribution networks has been introduced in literature such as PSO [3, 16], Harmony Search Algorithm (HSA) [17], Cuckoo Algorithm [18], chaotic firefly algorithm [19], differential evolution [20] and so on.

In this chapter, PSO is applied as a solution to the introduced power system operation problems, namely ED and DOCR coordination. The rest of the chapter is organized as follows: in Section 2, these power system problems are defined and formulated as optimization problems. PSO algorithm is explained in Section 3. In Section 4, PSO is applied in two case study systems to conduct the performance and feasibility of this method. Finally, Section 5 concludes the chapter with the results in pervious sections.

2. Problem formulation

In this section, overcurrent relay coordination and economic dispatch problems are formulated separately as optimization problems.

2.1. Relay coordination problem

In a protection scheme, each primary relay should operate as fast as possible to clear the fault in a system. If the operation time of the relay takes longer than an acceptable time, the damage on the faulty equipment would be severe with serious consequences. In other words, minimizing the total operation time of relays decreases the risk and stress on the protected apparatus, which can be depicted as an optimization objective function:

$$OF = \min \sum_{i=1}^{n} w_i t_i \tag{1}$$

where t_i is the operation time of relay R_i, w_i is the probability of the occurrence of fault on transmission line in the zone of protection, and it is normally set to 1; n is the total number of relays in the system.

Generally, the operation time of DOCRs is defined in (2):

$$t_i = \frac{\lambda \times TMS_i}{\left(\frac{I_{Fi}}{PS_i}\right)^{\eta} - 1} + L \tag{2}$$

where I_{Fi} is the fault current seen by the appropriate relay R_i after being transformed through the secondary winding of corresponding current transformer (CT). Depending on the type of relays, the characteristic constants λ, L and η are selected [16]. In this chapter, continuous form of TMS and PS is considered with relay type of standard inverse definite minimum time (IDMT). Based on that, all the relays in the system are assumed identical with a common characteristic function approximated by:

$$t_i = \frac{0.14 \times TMS_i}{\left(\frac{I_i}{PS_i}\right)^{0.02} - 1} \tag{3}$$

To ensure that the operation time of an individual relay is proper enough to mitigate the damage impact of faults on the apparatus, the time must be within an acceptable range:

$$t_{i\min} \le t_i \le t_{i\max}; i = 1, \ldots, n \tag{4}$$

where, respectively, $t_{i\min}$ and $t_{i\max}$ are the minimum and maximum operating time of the relay R_i. Each overcurrent relay has a manufactured TMS range to provide controllability of response to faults with different speeds. As shown in (3), t_i is proportional to TMS values. Also, the PS has nonlinear effect on the operating time. Within a security margin and to avoid maloperation of an individual relay with normal load or slight overload current, the minimum pickup current setting is selected bigger than the maximum load current. The maximum plug

setting is chosen not greater than the minimum fault current [19]. Therefore, there are constraints on TMS and PS as follows:

$$TMS_{i\min} \le TMS_i \le TMS_{i\max}; i = 1, \ldots, n \tag{5}$$

$$PS_{i\min} \le PS_i \le PS_{i\max}; i = 1, \ldots, n \tag{6}$$

where $TMS_{i\max}$, $PS_{i\max}$, $TMS_{i\min}$ and $PS_{i\min}$ are the maximum and minimum values of TMS and PS of relay R_i. Although the constraints in Eqs. (4)–(6) seem to provide satisfactory limits on performance of each relay, they are not enough to guarantee correct performance of the protection scheme as the coordination between primary-backup relays has not been considered.

The constraints can only ensure the operation of an individual relay not a primary-backup pair. To coordinate adjacent relays as primary and backup relays, the primary relay should operate as fast as possible within its acceptable boundaries. If it fails to act, its backup relay needs to take over the tripping action with a minimum time. The minimum operating time of the backup relay must be small but yet bigger than the operating time of primary relay. Therefore, a coordination time interval (CTI) is added to the constraints to satisfy the proper coordination scheme.

$$t_{j\ backup} - t_{i\ primary} \ge CTI \tag{7}$$

where t_i and t_j are the operation time of primary and backup relays, respectively. CTI depends upon the relay type, circuit breaker speed, relay over-travel time and the safety factor time for CT saturation, setting errors, contact gaps and so on. According to the IEEE standard [21], CTI is set to 0.2 s for the digital relays. **Figure 1** shows the backup-primary pair of relays in a radial network.

2.2. Economic dispatch

How to allocate the available generators in the grid to respond to the load demand with minimum fuel costs is an economical aspect of power dispatch in the electric power system that annually costs millions of dollars to operate.

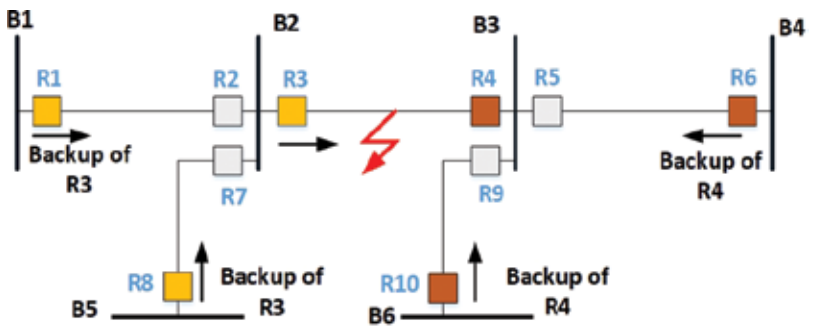

Figure 1. Primary-backup relation between relays in a distribution system.

In a practical ED optimization, the generator constraints and network limits such as ramp rate limit, the prohibited zones of operation, generation capacity constraints and valve point effects are considered. Single quadratic equation is used to formulate the ED optimization problem:

$$\min F_t = \sum_{i=1}^{m} F(P_i) = \sum_{i=1}^{m} \alpha_i + \beta_i P_i + \gamma_i P_i^2 \tag{8}$$

where α_i, β_i and γ_i are the cost equation coefficients of unit i, P_i is the output power in MW and $F(P_i)$ is the cost function of that unit in \$/h. The index m denotes the number of generators in a system.

The fuel cost function in (8) as the objective function of ED problem here is associated with practical constraints. Considering the ramp rate limits, the momentary output power of a generator cannot exceed its previously generated power more than a certain amount of UR_i, the up-ramp rate limit and neither can it be less than that of the previously generated power by more than a certain amount of DR_i, the down-ramp rate limit of the generator. Therefore:

$$P_i - P_{i0} \le UR_i \tag{9}$$

$$P_{i0} - P_i \le DR_i \tag{10}$$

where P_i is the current power output and P_{i0} refers to the previous power output of generator i. UR_i and DR_i represent the up-ramp limit and down-ramp limit of the generator i, respectively, in which the current generated power has the following constraint in MW/t:

$$P_{i0} - DR_i \le P_i \le P_{i0} - UR_i \tag{11}$$

The input-output curves of the generation units have separate operation zones. The prohibited zones of operations are due to the operation of steam valve or the shaft bearing vibration of the generators. Therefore, the generated power is within the feasible zones of operation and outside the prohibited zones. For a generator i:

$$P_i \in \begin{cases} P_i^{min} \le P_i \le P_{i,1}^{l} \\ P_{i,j-1}^{u} \le P_i \le P_{i,j-1}^{l} \\ P_{i,n}^{u} \le P_i \le P_i^{max} \end{cases}, j = 2, 3, \ldots, n; i = l, \ldots, m \tag{12}$$

where j is the number of prohibited zones of operation for unit i, $P_{i,j}^{l}$ and $P_{i,j}^{u}$ are the lower and upper boundaries, respectively, of prohibited zone j of generator i, $P_i^{\min}$ and $P_i^{\max}$ are minimum and maximum power capacity, respectively, of a generator that produces. From (11) and (12), it can be deduced that the operational power of a generator must be within the constraint in (13) and also be out of the prohibited zones in (12):

$$\max\left(P_i^{\min}, P_{i0} - DR_i\right) \le P_i \le \min\left(P_i^{\max}, P_{i0} + UR_i\right) \tag{13}$$

Total delivered power from the units needs to meet the power demand and transmission loss in the grid.

$$\sum_{i=1}^{m} P_i = P_D + P_L; i \in m \tag{14}$$

where P_D and P_L represent the demand power and the power loss in the gird, respectively. The overall power loss of the committed units is based on the output power, which is formulated by B matrix coefficients known as Kron's formula:

$$P_L = \sum_{i=1}^{m} \sum_{j=1}^{m} P_i B_{ij} P_j + \sum_{i=1}^{m} B_{0i} P_i + B_{00} \tag{15}$$

The power loss itself cannot be more than some permissible values:

$$\left|P_{Lf,k}\right| \leq P_{Lf,k}^{\max} \; ; k = j, \ldots, L \tag{16}$$

where the real power flow of line j is represented with $P_{Lf,k}$ and k is the number of transmission lines in a system. The power loss cannot be more than a maximum value of $P_{Lf,k}^{\max}$.

3. Particle swarm optimization algorithm

PSO algorithm is a nature inspired method from social behavior of bird flocking and fish schooling, which is first introduced by Erberhart and Kennedy in 1995 [22]. As a population-based stochastic optimization technique using swarm intellects in the search space, this technique is based on interaction of swarm of particles. Every particle includes two values of position and velocity that are updated during the iteration runs by considering each particle's best experience (best position) and the best achieved experience (global position) of all particles.

The update of position and velocity of the particles must be processed, and it has to follow Eqs. (17)–(19) for satisfying the constraint of an optimization problem. Each particle movement is based on the changes in its position and its velocity:

$$p_i^{k+1} = p_i^k + v_i^{k+1} \tag{17}$$

where p_i^{k+1} and p_i^k are the position of particle i in the iteration $k+1$ and k, respectively, v_i^{k+1} is the velocity of the particle in $k+1$ iteration. A particle's velocity is defined as follows:

$$v_i{}^{k+1} = w \times v_i{}^k + c_1 \times rand_1 \times \left(pbest_i - p_i{}^k\right) + c_2 \times rand_2 \times \left(gbest - p_i{}^k\right) \tag{18}$$

where $pbest_i$ is the best so far position of the particle i as the best experience, while $gbest$ is the best position among the whole swarm with all the particles in movement as global experience. c_1 and c_2 are the weighting factors, while $rand_1$ and $rand_2$ are two random numbers between zero and one. The parameter w is the inertia factor varying between $[w_{\min}, w_{\max}]$, as shown in (19), which is a linear decreasing inertia weight in this chapter.

$$w = w_{\max} - \frac{w_{max} - w_{\min}}{iter_{\max}} \times k \tag{19}$$

where k and $iter_{\max}$ are the current iteration and the maximum number of iteration during simulations, respectively.

The general steps of PSO on solving an optimization problem are as follows:

1. Set initial parameters $w_{\min}$, $w_{\max}$, c_1 and c_2.

2. Generate initial populations having initial positions p and velocities v.

3. Set iteration $k = 1$.

4. Calculate fitness of particles $F_i^k = f\left(p_i^k\right), \forall i$ and find the index of the best particle b.

5. Select $pbest_i^k = p_i^k, \forall i$ and set $gbest^k = p_b^k$.

6. Update inertia factor: $w = w_{\max} - \frac{w_{max} - w_{\min}}{iter_{\max}} \times k$.

7. Update the velocity and position of particles.

$$v_i^{k+1} = w \times v_i^k + c_1 \times rand_1 \times \left(pbest_i - p_i^k\right) + c_2 \times rand_2 \times \left(gbest - p_i^k\right); \forall i$$
$$p_i^{k+1} = p_i^k + v_i^{k+1}; \forall i$$

8. Calculate fitness $F_i^{k+1} = f\left(p_i^{k+1}\right), \forall i$ and obtain the index of the best particle b_1.

9. Update *pbest* for all particles.

$$\text{if } F_i^{k+1} < F_i^k \text{ then } pbest_i^{k+1} = p_i^{k+1} \text{ else } pbest_i^{k+1} = pbest_i^k$$

10. Update *gbest* of the population

$$\text{if } F_{b1}^{k+1} < F_b^k \text{ then } gbest^{k+1} = pbest_{b1}^{k+1} \text{ and } b = b1 \text{ else } gbest^{k+1} = gbest^k$$

11. If $k < Iter_{\max}$ then $k = k + 1$ and go to step 6 else go to step 12.

12. Print $gbest^k$ as optimum solution.

4. Simulation results

Application of PSO on solving the defined problems in previous sections is validated here. In the first case study system, the overcurrent relay coordination problem in a distribution network is solved. The size of population and iteration numbers of PSO are 60 and 100, respectively. In the second case study system, the ED problem is addressed with population size of 100 and 500 iterations. c_1, c_2, $w_{\min}$ and $w_{\max}$ in both case study systems are the same and set to 2, 2, 0.4 and 0.9, respectively.

4.1. Case study 1: relay protection coordination

A 15-node radial network system including a total of 13 loads is considered, and PSO is employed to determine the optimal settings of all 28 digital over-current relays shown in **Figure 2**. Identical digital relays are used with same current transformer (CT) ratio 500:1. The constraint values of the relays are as follows: t_{imin} and t_{imax} for each relay is set to 0.1 and 4 s, respectively; TMS_{min} and TMS_{max} constraints are 0.1 and 1.1, respectively; PS_{min} and PS_{max} are 0.5 and 2.5, respectively [16].

The primary-backup relationships of the relays in the system are shown in **Table 1**. According to the objective function in (3), the maximum fault currents sensed by the relays are also required. Therefore, the case study system in **Figure 2** has been modeled in DigSILENT PowerFactory software with simulating three-phase faults occurring in front of each relay. The collected data have been shown in **Table 2**. The load parameters of the radial network are shown in **Table 3**.

After data collection in DigSILENT PowerFactory software, the optimal settings (TMS and PS) of all 28 overcurrent relays are obtained by solving (3) subject to the constraints in (4)–(7) using the PSO in MATLAB software. The algorithm was executed 100 times to achieve accurate results. **Table 4** shows the best TMS and PS settings of the relays to provide a reliable protection scheme

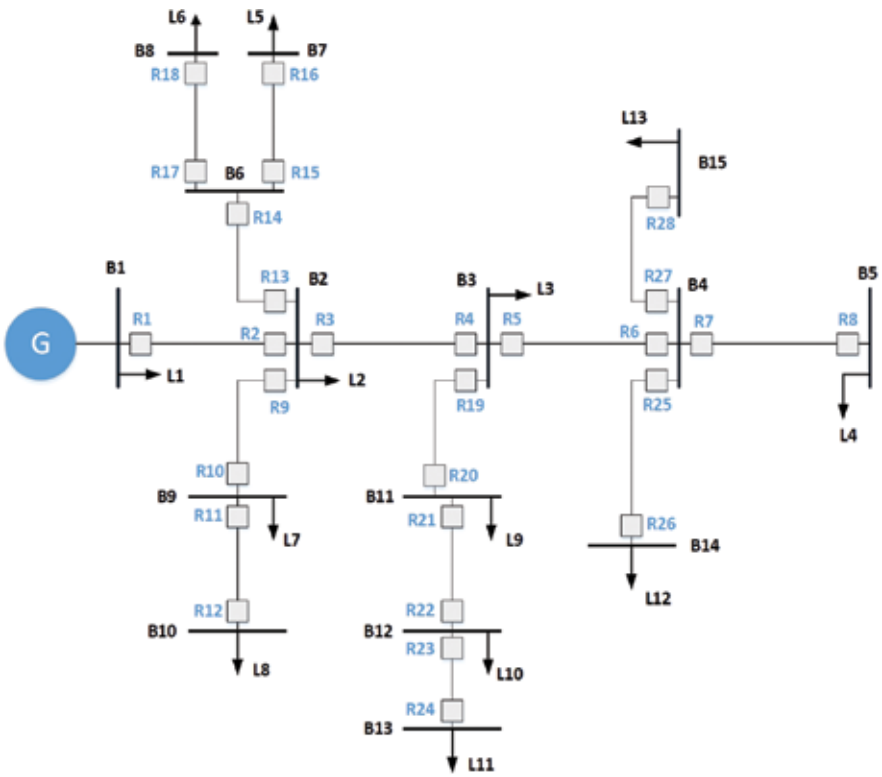

Figure 2. A 15-node distribution system as case study one.

Primary	Backup	Primary	Backup	Primary	Backup	Primary	Backup
1	—	8	—	15	13	22	—
2	—	9	1	16	—	23	21
3	1	10	—	17	13	24	—
4	—	11	9	18	—	25	5
5	3	12	—	19	3	26	—
6	—	13	1	20	—	27	5
7	5	14	—	21	19	28	—

Table 1. Primary-backup pair relationship of the relays.

Relay	Fault current (kA)	Relay	Fault current (kA)	Relay	Fault current (kA)	Relay	Fault current (kA)
1	22.778	8	4.636	15	7.7	22	4.636
2	11.532	9	11.509	16	5.79	23	4.632
3	11.509	10	7.71	17	7.7	24	3.866
4	7.71	11	7.7	18	5.79	25	5.785
5	7.7	12	5.79	19	7.7	26	4.636
6	5.79	13	11.509	20	5.79	27	5.785
7	5.785	14	7.71	21	5.785	28	4.636

Table 2. Maximum fault currents.

Load	Active power (MW)	Reactive power (Mvar)	Load	Active power (MW)	Reactive power (Mvar)
1	4	1.5	8	0.5	0.2
2	1	0.2	9	2	0.8
3	2.5	1	10	1	1
4	3	2	11	2.5	0.9
5	1	0.5	12	1	0.5
6	3	0.7	13	3	2
7	1	0.5			

Table 3. Load parameters in case study one.

in the distribution system. The total operation time of the relays in this system is 26.189 s. **Figure 3** illustrates the convergence curve of PSO in solving the objective function.

4.2. Case study 2: economic dispatch

A 15-unit test system is used to investigate the feasibility of PSO in solving the nonsmooth economic dispatch considering transmission losses, ramp rate limits and the prohibited

Relay	TMS	PS	Relay	TMS	PS
1	0.74394	0.54999	15	0.42632	0.79222
2	0.32216	1.2551	16	0.54396	0.67839
3	0.3416	1.5153	17	0.21794	0.61987
4	0.55245	2.4141	18	0.12037	1.1896
5	0.36342	0.60859	19	0.31997	0.66671
6	0.4309	1.0805	20	0.56566	1.1323
7	0.11188	1.2611	21	0.26376	0.60754
8	0.2726	1.822	22	0.14313	1.8741
9	0.45395	1.1148	23	0.12496	0.7978
10	0.15359	1.3933	24	0.55302	1.3848
11	0.32133	0.50089	25	0.22254	0.65529
12	0.26418	2.4727	26	0.2904	1.6316
13	0.32088	1.9075	27	0.13471	0.77885
14	0.73899	2.1497	28	0.10277	2.4925
OF (s)	26.189				

Table 4. Obtained TMS and PS values of relays by PSO.

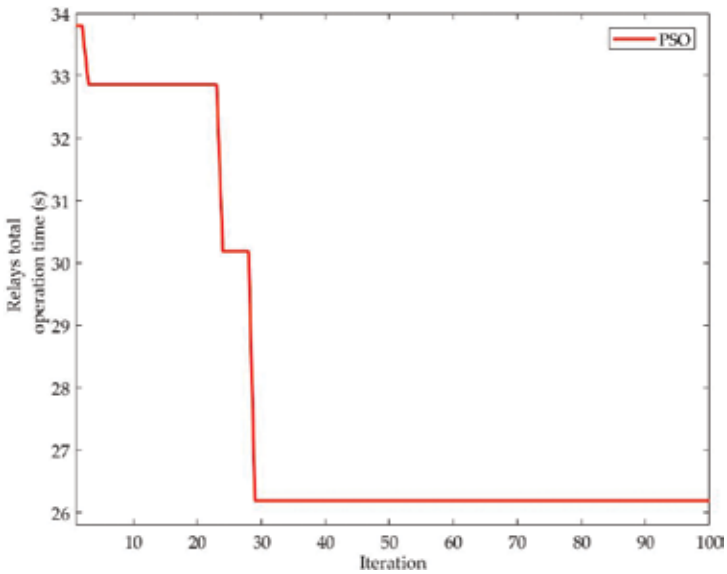

Figure 3. Convergence of PSO solution in case study one.

operation zones of the thermal generators. The cost curves data and operation limits of the 15-unit system are shown in **Table 5**. The B loss coefficients to calculate the power loss can be found in [4].

Unit	$P_i^{\min}$	$P_i^{\max}$	P_{i0}	α_i	β_i	γ_i	UR_i	DR_i	Prohibited zones (MW)
1	150	455	400	671	10.1	0.000299	80	120	
2	150	455	300	574	10.2	0.000183	80	120	[185 225] [305 335] [420 450]
3	20	130	105	374	8.80	0.001126	130	130	
4	20	130	100	374	8.80	0.001126	130	130	
5	150	470	90	461	10.40	0.000205	80	120	[180 200] [305 335] [390 420]
6	135	460	400	630	10.10	0.000301	80	120	[230 255] [365 395] [430 455]
7	135	465	350	548	9.8	0.000364	80	120	
8	60	300	95	227	11.2	0.000338	65	100	
9	25	162	105	173	11.2	0.000807	60	100	
10	25	160	110	175	10.7	0.001203	60	100	
11	20	80	60	186	10.2	0.003586	80	80	
12	20	80	40	230	9.90	0.005513	80	80	[30 40] [55 65]
13	25	85	30	225	13.1	0.000371	80	80	
14	15	55	20	309	12.1	0.001929	55	55	
15	15	55	20	323	12.4	0.004447	55	55	

Table 5. Generation unit characteristics of a 15-unit system in case study two.

This system has many local minima with high dimensionality that draws realistic analysis for practical applications. PSO is applied on the 15-unit test system, and the results are compared with best results in literature on the same system: ACHS [8], SQPSO [11], ABC [6], IPSO [12], PSO-MSAF [13], DSPSO-TSA [14], ICA [5], GAAPI [9], MPSO [10], SOH-PSO [15], GCPSO [10], PC-PSO [15], BF [7], SPSO [15], PSO [4] and GA [4].

The simulation is tested for 100 times to ensure reliable analysis. **Table 6** shows the optimum results of each method for the 15-unit system. **Figure 4** illustrates the convergence of PSO method in 100 different trials, while the final fuel costs in 100 trials are shown in **Figure 5**. **Table 7** shows the best economic dispatch of power using PSO for ED optimization in the 15-unit system.

The optimum cost of 15-unit system with the proposed PSO solution is 32701.282 ($), which has better result than other methods. GA has the most deviating results compared with other hybrid and improved methods in this test system. Also, the mean value of final fuel cost of generators using PSO over 100 trials is less than minimum values obtained by other method, which indicates higher quality of solution and better performance of PSO compared with other algorithms on the same test system. The power loss in the grid obtained from PSO is less than other algorithms, which shows better dispatching scheme using PSO. The best convergence of PSO to the minimum fuel cost of generators in the grid while satisfying the constraints is shown in **Figure 6**.

In case study two, PSO achieves better results when compared with other hybrid or improved methods. It is worth mentioning that the maximum iteration number is 500 and the population

Method	Best cost ($/h)	Mean ($/h)
PSO	32701.282	32704.10578
ACHS [8]	32706.6500	32706.65
SQPSO [11]	32706.6740	32708.4457
ABC [6]	32707.85	32707.95
IPSO [12]	32709.00	32784.5
PSO-MSAF [13]	32713.09	32759.64
DSPSO-TSA [14]	32715.06	32724.63
ICA [5]	32715.4305	NA
GAAPI [9]	32732.95	NA
MPSO [10]	32738.4177	NA
SOH-PSO [15]	32751.39	32,878
GCPSO [10]	32764.4616	NA
PC-PSO [15]	32775.36	NA
BF [7]	32784.5024	32796.81
SPSO [15]	32798.69	NA
PSO [4]	32,858	33,039
GA [4]	33,113	33,228

Table 6. Comparison of PSO results with other methods in case study two.

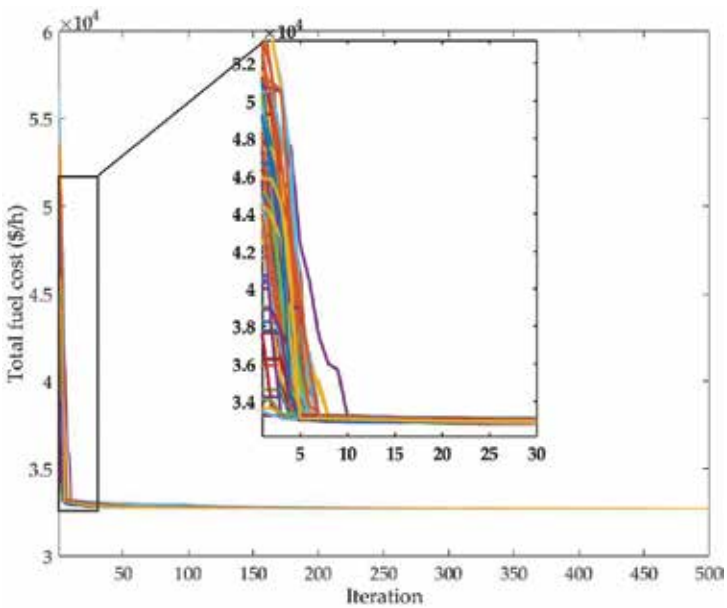

Figure 4. Convergence curve of PSO over 100 different trials.

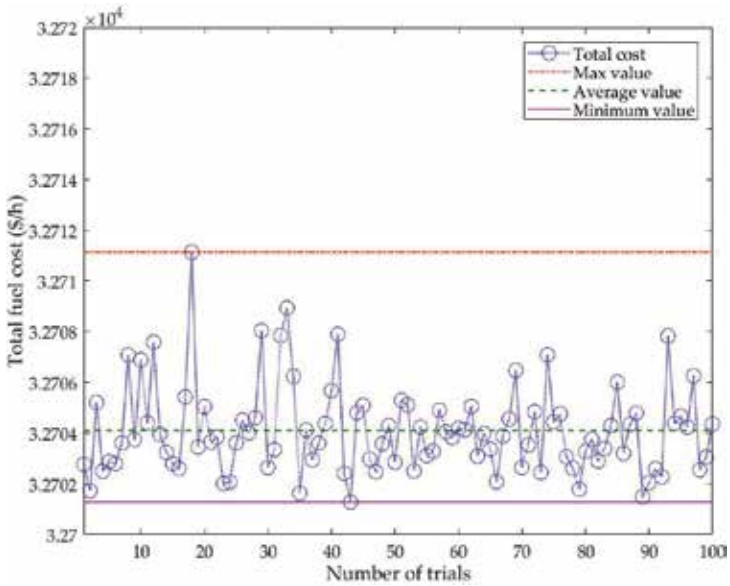

Figure 5. Cost distribution with PSO over 100 different trials.

Unit (MW)	PSO	ICA [5]	GCPSO [10]	MPSO [10]	GA [4]	PSO [4]	ABC [6]	SOH-PSO [15]
P1	454.9963	455	449.89252	455	415.3108	439.1162	455	455
P2	379.9998	380	366.99066	380	359.7206	407.9727	380	380
P3	130	130	130	130	104.4250	119.6324	130	130
P4	129.9954	130	130	130	74.9853	129.9925	130	130
P5	169.9999	167.4174	170	170	380.2844	151.0681	169.9997	170
P6	459.9999	460	460	460	426.7902	459.9978	460	459.96
P7	430	430	430	430	341.3164	425.5601	430	430.00
P8	66.1794	113.4737	75.88460	92.7278	124.7867	98.5699	71.9698	117.53
P9	64.9485	25.1555	50.22689	43.0282	133.1445	113.4936	59.1798	77.90
P10	159.2255	155.3478	160	140.1938	89.2567	101.1142	159.8004	119.54
P11	79.9996	80	80	80	60.0572	33.9116	80	54.50
P12	79.9901	80	77.87063	80	49.9998	79.9583	80	80.00
P13	25.0001	25	25	27.6403	38.7713	25.0042	25.0024	25.00
P14	15.0005	15	15.8312	20.7610	41.9425	41.4140	15.0056	17.86
P15	15.0029	15	39.66146	22.2724	22.6445	35.6140	15.0014	15
Total power	2660.338	2661.394	2661.35806	2661.6235	2668.4	2262.4	2735.959	2662.29
Power loss	30.338	31.291	30.86593	29.978	38.2782	32.4306	30.9591	32.28
Cost ($)	32701.282	32715.4305	32764.4616	32738.41778	33113	32858	32707.85	32751.39

Table 7. Optimum solution of PSO compared with other methods in case study two.

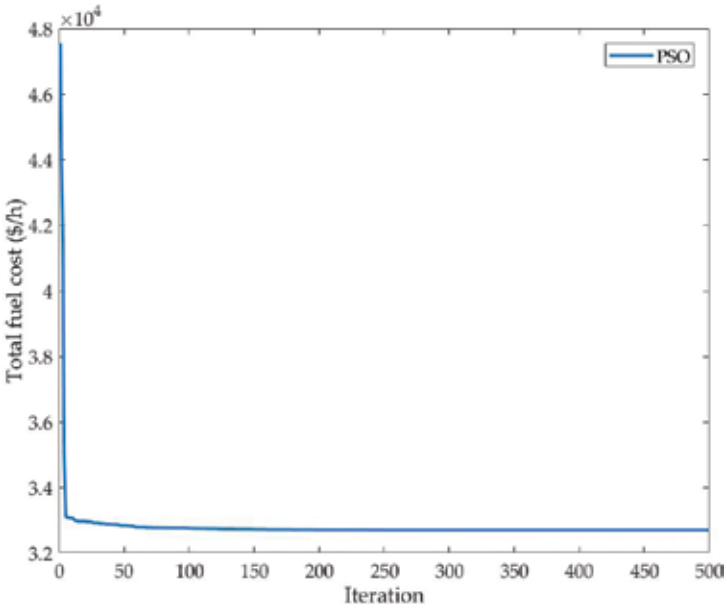

Figure 6. Convergence of PSO with best fuel cost result for ED problem in case study two.

size is 100. In most of the quoted methods, the iteration and population sizes vary that can affect the final results. For example, the PSO in [4] has population size and iteration number of 100 and 500, respectively. The population size, iteration, crossover rate, mute rate and crossover parameter of GA [4] are 100, 200, 0.8, 0.01 and 0.5, respectively.

Different system configuration and programming language frameworks can also influence the results in which MATLAB 2015Ra was used for programming.

5. Conclusions

Distribution network relay coordination and the economic dispatch of generators in the electric power system were modeled as optimization problems. Particle swarm optimization (PSO) was successfully employed to solve the defined problems where two case study systems were conducted to validate the results. In the first case study system, PSO provided proper relay settings that allow all the relays in a system to perform with high reliability and accuracy. In the second case, the optimal power outputs of thermal generators in the grid were scheduled to satisfy the load demands and other practical constraints on the generators and the grid with minimum fuel costs. The compared results with other methods demonstrated higher quality of solution, and less fuel costs obtained by PSO. The general performance of PSO in this chapter indicates applicability of this method on practical power system-related problems that are difficult to be handled by conventional methods.

Author details

Mostafa Kheshti* and Lei Ding

*Address all correspondence to: mostafa_kheshti@yahoo.com

Key Laboratory of Power System Intelligent Dispatch and Control, Shandong University, Ministry of Education, Jinan, China

References

[1] Kheshti M, Kang X, Bie Z, Jiao Z, Wang X. An effective lightning flash algorithm solution to large scale non-convex economic dispatch with valve-point and multiple fuel options on generation units. Energy. 2017;**129**:1-15

[2] Kheshti M, Kang X, Li J, Regulski P, Terzija V. Lightning flash algorithm for solving nonconvex combined emission economic dispatch with generator constraints. IET Generation, Transmission & Distribution. 2017. DOI: 10.1049/iet-gtd.2017.0257

[3] Kheshti M, Kang X, Jiao Z. An innovative fast relay coordination method to bypass the time consumption of optimization algorithms in relay protection coordination. Journal of Electrical Engineering & Technology. 2017;**12**(2):612-620

[4] Gaing ZL. Particle swarm optimization to solving the economic dispatch considering the generator constraints. IEEE Transactions on Power Systems. 2003;**18**(3):1187-1195

[5] Bijami E, Jadidoleslam M, Ebrahimi A, Askari J, Farsangi MM. Implementation of imperialist competitive algorithm to solve non-convex economic dispatch problem. Journal of the Chinese Institute of Engineers. 2014;**37**(2):232-242

[6] Hemamalini S, Simon SP. Artificial bee colony algorithm for economic load dispatch problem with non-smooth cost functions. Electric Power Components and Systems. 2010;**38**(7):786-803

[7] Panigrahi BK, Pandi VR. Bacterial foraging optimisation: Nelder-mead hybrid algorithm for economic load dispatch. IET Generation, Transmission & Distribution. 2008;**2**(4):556-565

[8] Niu Q, Zhang H, Wang X, Li K, Irwin GW. A hybrid harmony search with arithmetic crossover operation for economic dispatch. International Journal of Electrical Power & Energy Systems. 2014;**62**(1):237-257

[9] Ciornei I, Kyriakides E. A GA-API solution for the economic dispatch of generation in power system operation. IEEE Transactions on Power Systems. 2012;**27**(1):233-242

[10] Neyestani M, Farsangi MM, Nezamabadi-pour HA. Modified particle swarm optimization for economic dispatch with non-smooth cost functions. Engineering Applications of Artificial Intelligence. 2010;**23**(7):1121-1126

[11] Hosseinnezhad V, Rafiee M, Ahmadian M, Ameli MT. Species-based quantum particle swarm optimization for economic load dispatch. International Journal of Electrical Power & Energy Systems. 2014;**63**(1):331-322

[12] Safari A, Shayegui H. Iteration particle swarm optimization procedure for economic load dispatch with generator constraints. Expert Systems with Applications. 2011;**38**(5):6043-6048

[13] Subbaraj P, Rengaraj R, Salivahanan S, Senthilkumar TR. Parallel particle swarm optimization with modified stochastic acceleration factors for solving large scale economic dispatch problem. International Journal of Electrical Power & Energy Systems. 2010;**32**(9):1014-1023

[14] Khamsawang S, Jiriwibhakorn S. DSPSO–TSA for economic dispatch problem with nonsmooth and noncontinuous cost functions. Energy Conversion and Management. 2010;**51**(2):365-375

[15] Chaturvedi KT, Pandit M, Srivastava L. Self-organizing hierarchical particle swarm optimization for nonconvex economic dispatch. IEEE Transactions on Power Systems. 2008; **23**(3):1079-1087

[16] Kheshti M, Tekpeti BS, Kang X. The optimal coordination of overcurrent relay protection in radial network based on particle swarm optimization. In: 2016 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC); 25-28 Oct. 2016; Xi'an, China. IEEE; 2016. p. 604-608

[17] Rajput VN, Pandya KS. Coordination of directional overcurrent relays in the interconnected power systems using effective tuning of harmony search algorithm. Sustainable Computing: Informatics and Systems. 2017;**15**:1-15

[18] Ahmarinejad A, Hasanpour SM, Babaei M, Tabrizian M. Optimal overcurrent relays coordination in microgrid using cuckoo algorithm. Energy Procedia. 2016;**100**:280-286

[19] Gokhale SS, Kale VS. An application of a tent map initiated chaotic firefly algorithm for optimal overcurrent relay coordination. International Journal of Electrical Power & Energy Systems. 2016;**78**:336-342

[20] Shih MY, Enríquez AC, Hsiao TY, Treviño LMT. Enhanced differential evolution algorithm for coordination of directional overcurrent relays. Electric Power Systems Research. 2017;**143**:365-375

[21] IEEE Recommended Practice for Protection and Coordination of Industrial Power Systems. IEEE Std. 242-2001, December 2001

[22] Kennedy J, Eberhart RC. Particle swarm optimization. In: 4th IEEE International Conference on Neural Networks (ICNN); 27 November-01 December; Perth, Australia. 1995. p. 1942-1948

Chapter 4

Stochastic Greedy-Based Particle Swarm Optimization for Workflow Application in Grid

Ruey-Maw Chen and Yin-Mou Shen

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.73587

Abstract

The workflow application is a common grid application. The objective of a workflow application is to complete all the tasks within the shortest time, i.e., minimal makespan. A job scheduler with a high-efficient scheduling algorithm is required to solve workflow scheduling based on grid information. Scheduling problems are NP-complete problems, which have been well solved by metaheuristic algorithms. To attain effective solutions to workflow application, an algorithm named the stochastic greedy PSO (SGPSO) is proposed to solve workflow scheduling; a new velocity update rule based on stochastic greedy is suggested. Restated, a stochastic greedy-driven search guidance is provided to particles. Meanwhile, a stochastic greedy probability (SGP) parameter is designed to help control whether the search behavior of particles is exploitation or exploration to improve search efficiency. The advantages of the proposed scheme are retaining exploration capability during a search, reducing complexity and computation time, and easy to implement. Retaining exploration capability during a search prevents particles from getting trapped on local optimums. Additionally, the diversity of the proposed SGPSO is verified and analyzed. The experimental results demonstrate that the SGPSO proposed can effectively solve workflow class problems encountered in the grid environment.

Keywords: scheduling, optimization, stochastic greedy, workflow, particle swarm optimization

1. Introduction

Grid computing is applied mainly to utilize the heterogeneous computational resources to execute various applications. The computing ability of the grid is comparable to that of a super computer, and the grid computing environment consists of heterogeneous computing devices throughout the world and connected by low-latency and high-bandwidth networks [1]. The

main application of the grids is the sharing of computing resources. Scholars have already used the grid in real cases when requiring vast computation and immense storage space [1–3]. The basic scenario of grid computing application is shown in **Figure 1**. The job scheduler uses grid information supplied by information server which collects grid information from grid sensors. When a workflow application comprising numerous tasks with partially ordered constraint is uploaded to the grid, the job scheduler of the grid platform allocates the tasks to be processed to the computing resources on the grid. If the tasks and the resources are well scheduled, the time needed to complete all the tasks of the workflow application can then be reduced. Otherwise, the time will be extended. Restated, the makespan of workflow application on the grid is highly impacted by the quality of task-resource arrangement. Many workflow application scheduling algorithms have been presented to boost efficiency and make the resource manager more efficient when matching tasks and resources so that grid performance can be upgraded effectively.

Many studies have developed scheduling optimization methods intended to reduce the makespan of jobs (all tasks) on the grid [4, 5–9]. When restrictions regarding partially ordered tasks exist between tasks (i.e., dependent tasks), the algorithm applied must meet the needs of such sequential relationships when scheduling optimization is conducted. Besides solving the task-resource matching problem, the sequence of execution of independent tasks allocated to the same resource also has a rather significant effect on the reduction of the makespans of jobs in workflow scheduling. In other words, workflow scheduling has to simultaneously solve two subproblems in task-resource matching and those unrelated to tasks' priorities. Most scheduling problems are NP-complete, and many heuristic and metaheuristic algorithms have been proposed to solve NP problems, such as the ant colony optimization (ACO) [3], genetic algorithm (GA) [6], simulated annealing (SA) [5], and particle swarm optimization (PSO) [10]. Among them, PSO carries the advantages of easy implementation, requiring fewer parameters and having a faster convergence speed; therefore, PSO is often used to solve scheduling problems in fields aside from a grid or cloud computing, such as course timetabling problems [11], flowshop problems [12], and vehicle routing problems (VRP) [13]. Also, PSO was applied

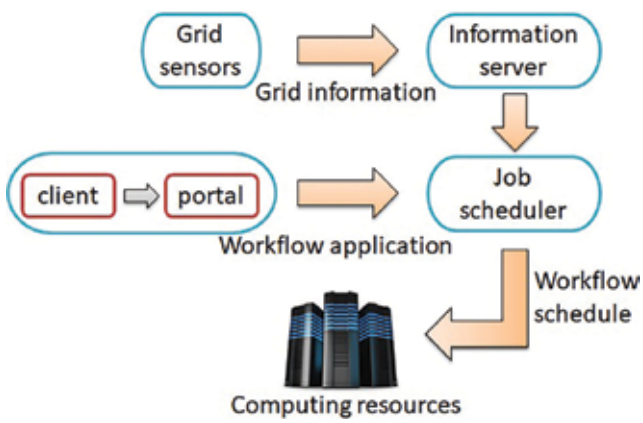

Figure 1. Grid application scenario.

to solve grid scheduling problems; Tao et al. [14] and Chen and Wang [15] have all adopted the PSO to solve grid scheduling problems effectively.

Two subproblems have to be dealt with in workflow scheduling: the task-resource mapping and the execution priorities of tasks without precedence constraint in the same computing resource. To these two subproblems, two PSO algorithms were designed to find the corresponding solutions: a discrete-type PSO algorithm to solve the task-resource mapping subproblem and a constriction-type PSO algorithm to solve the task priority subproblem. In PSO, the location of each particle represents a solution to the problem to be solved, and each particle moves with reference to global experience and individual experience, resulting in a new solution. In this study, a new velocity update rule developed from state transformation rules used in ant colony optimization (ACO) [16] was proposed rather than the velocity update rule in the contraction-type PSO, where movement is based on both experiences. This new PSO is named the stochastic greedy PSO (SGPSO) herein. In ACO, the ant moves by referencing the highest pheromone. Besides movement guided by the highest pheromone trail, the ant also references the other trail, determined using the roulette wheel rule to move. In this study, the global experience of the particle swarm is regarded as the path of ants with the highest pheromone. Thus, a new velocity update rule was introduced to allow the particles with the probability to explore on their own in the solution search process and prevent them from getting trapped on local optimums. Restated, the particle moves in accordance with either the global experience or individual experience in this work. Moreover, different problem cases with small-scale and large-scale problems are designed and tested to verify the performance of the proposed SGPSO. In the end, the workflow scheduling problems in [15] were tested, and comparative analysis was conducted. Furthermore, the diversity of the proposed SGPSO is also defined and verified. The remaining parts of this paper are arranged as follows: Section 2 presents the workflow scheduling problems, Section 3 describes the new velocity update rule applied in the particle swarm optimization algorithm and the concept behind its adoption from ACO, in Section 4 the experimental results and comparative analysis are provided, and the conclusions are presented in Section 5.

2. Description of workflow scheduling problems

In the grid environment, distributed computing is conducted with the resources that are scattered among different places around the world and connected together through networks. The composition of the grid environment is shown in **Figure 2(a)**. The scattered computational resources are linked through the Internet. Each resource has its own computing ability and external bandwidth represented by AB_u and BW_u, respectively. Moreover, the resources in the grid environment are heterogeneous resources, meaning that the computing ability and external bandwidth of each resource are dissimilar. Generally, the grid environment can be represented with a schematic $G(R, C)$ composed of nodes and edges. Each node stands for a resource ($\boldsymbol{R} = \{R_i\}$). i = 1 ~ N represents the set of all resources. N is the total amount of resources in the grid environment. The connections between resources are represented by an edge. $\boldsymbol{C} = \{C_{uv}\}$ stands for the set of resource-resource connections. C_{uv} is the connection between

resource u and resource v. The grid computing environmental schematic is shown in **Figure 2 (b)**. The workflow application can be represented with a directed acyclic graph (DAG) $G(\boldsymbol{V}, \boldsymbol{T})$, as shown in **Figure 3**, in which $\boldsymbol{V} = \{V_i\}$, where $i = 1 \sim M$ is the set of all tasks and M is the total number of tasks. Precedence exists between certain tasks, and each task has a workload; w_i represents the workload of task i. **Figure 3** also indicates the precedence relationship between tasks such as the following: task 1 is the predecessor of tasks 3 and 4, and tasks 3 and 4 are the successors of task 1. Meanwhile, task 5 cannot be executed until task 2 is done, and task 6 has to wait till tasks 3, 4, and 5 are accomplished. Certain required data for execution on successors have to be transmitted to the successors when the predecessor is completed, i.e., transmission costs exist. TC_{ij} represents the amount of data transmitted between tasks i and j. If the predecessor and the successors are arranged to be executed with different resources, a transmission cost exists. On the contrary, if the predecessor and the successors are arranged to be executed with the same resource, there will be no transmission cost. Meanwhile, task 0 and task 8 in the figure are virtual tasks representing the start and the end, respectively. They have no workload and involve no data transmission costs.

The goals of workflow scheduling optimization are to appropriately match tasks to resources and to suitably assign execution priorities to tasks without precedence restriction in the same computing resource to reduce the makespan of the application execution. The cost includes the resource processing time of the path and the data transmission time. In **Figure 3**, for example,

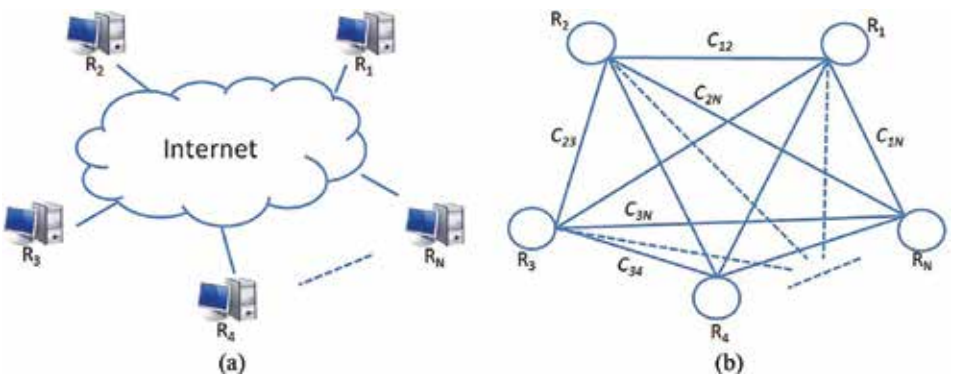

Figure 2. Grid computing environment. (a) The composition of the grid environment. (b) The grid computing environmental schematic.

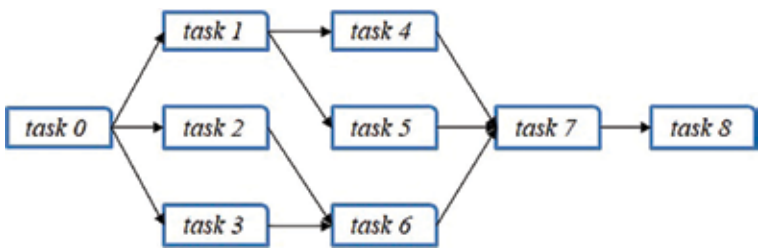

Figure 3. Directed acyclic graph (DAG) of a workflow application on the grid.

there are four execution sequence paths: (p1) task 0→task 1→task 4→task 7→task 8, (p2) task 0→task 1→task 5→task 7→task 8, (p3) task 0→task 2→task 6→task 7→task 8, and (p4) task 0→task 3→task 6→task 7→task 8. All the tasks in the four execution sequences must be executed to complete the job on the grid. The makespan is subject to the time of the longest execution sequence path. That is, the max($cost(p_i)$). $cost(p_i)$ is the cost of an execution sequence path (p_i) in DAG in the grid environment.

Calculation of $cost(p_i)$ is shown in Eq. (1). $cost(p_i)$ is the aggregate of the total resource processing time on execution sequence path p_i and the total data transmission time on that path. In Eq. (1), $u(w_t)$ represents the workload of the tasks allocated to resource u, and $cost(tf)$ is the data transmission cost or time on that execution sequence path, as shown in Eq. (2):

$$cost(p_i) = \frac{\sum_{u \in p_i} u(w_t)}{AB_u} + \sum_{tf \in \mathbf{T}} cost(tf) \tag{1}$$

$$cost(tf) = \frac{TC_{ij}}{\min\{BW_u, BW_v\}} \tag{2}$$

If task i and task j are, respectively, allocated to be executed with the resources u and v, between the two tasks exists the amount of data transmission ($TCij$). Since resource u and resource v have different bandwidths, the data transmission time is subject to the smaller bandwidth ($BW_{uv} = \min\{BW_u, BW_v\}$). Hence, the data transmission time or cost is the amount of data transmitted divided by the smaller resource bandwidth.

Therefore, the makespan is defined as the fitness function to denote the quality of workflow scheduling. The definition of fitness function (FIT) is shown in Eq. (3). The objective of workflow scheduling is then to find the shortest makespan (min(FIT)):

$$FIT = \max\{\cos t(p_i) | p_i \in DAG\}, \tag{3}$$

3. The proposed method

Many nature-inspired optimization algorithms have been proposed to find optimal solutions to workflow scheduling problems and metaheuristic algorithms that imitate the behaviors of biological creatures. Some that are extensively applied include ACO, GA, bee colony optimization (BCO), and the PSO adopted in this study. Among them, PSO requires fewer parameters and is easier to implement. Therefore, it has been well applied to solve diverse *NP*-complete problems, and the results have been rather remarkable. Meanwhile, PSO has also been employed to solve workflow scheduling problems with effectiveness.

As shown in **Figure 3**, if task 2 and task 6 are allocated to be computed by different resources, there will be a transmission time, and the makespan will be extended. On the contrary, if they are arranged to be executed by the same resource, there will be no transmission time, and the makespan will be shortened. Furthermore, if task 1 and task 2 are arranged to be executed by the same resource, executing task 1 before task 2 or vice versa will have an effect on the

makespan of the workflow application on the grid. Hence, the study of minimization of the makespan in workflow scheduling involves two issues: task-resource matching and task execution priority determination. **Figure 4** illustrates an example with two heterogeneous resources (R_1 and R_2) with different abilities in the grid environment; these two subproblems need to be solved for the studied workflow application scheduling problem. **Figure 4(a)** displays the task-resource matching subproblem, and **Figure 4(b)** indicates some possible execution priorities of the tasks (tasks 1, 3, 4, and 5) assigned to resource 1 (R_1).

To deal with these issues, the constriction PSO is used to solve the task execution priority issue and the discrete PSO to cope with the task-resource matching issue.

3.1. Used PSOs

PSO was first introduced by Kennedy and Ebrhart in 1995 [17]. After observing birds flying, fish seeking food, and other social behaviors of animals, they discovered that each particle would move to their next position according to information (experience) shared among the members of the swarm. Restated, when particles move, they refer to individual experience (*pbest*) and global experience (*gbest*). At each moving step, a particle will refer to both kinds of experience to make the next move. The particle represents the solution of the problem to be solved. Hence, the movement of particles is regarded as a solution search in a solution space. The position update equation of PSO is as shown in Eq. (4):

$$\begin{cases} V_{ij}^{new} = \omega V_{ij} + c_1 r_1 \left(pbest_{ij} - X_{ij} \right) + c_2 r_2 \left(gbest_j - X_{ij} \right) \\ X_{ij}^{new} = X_{ij} + V_{ij}^{new} \end{cases} \tag{4}$$

Each particle of a swarm has its own velocity *V*. The V_{ij} represents the *j*th velocity component of the particle *i*. X is the position of the particle and X_{ij} the *j*th position component of particle *i*. $pbest_{ij}$ is the *j*th component of the best individual experience of the particle *i*. The $gbest_j$ is the *j*th component of the best global experience. *w* is the inertia weight adopted mainly for controlling the level of influence of the previous velocity on the velocity of the current iteration. c_1 and c_2 are the learning factors for controlling the influence of individual best experience and global

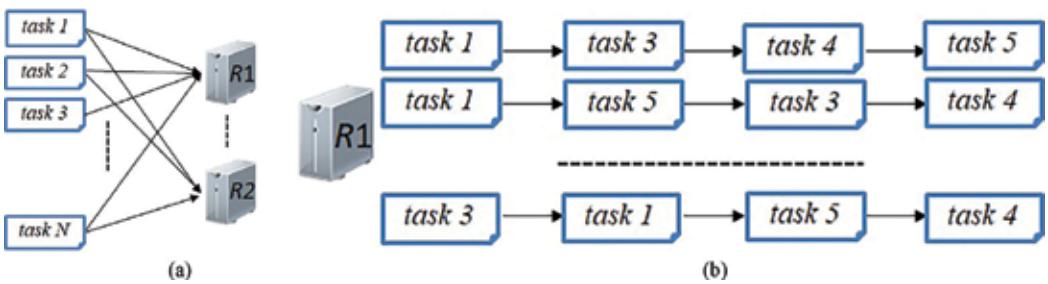

Figure 4. Two subproblems of a workflow application in grid. (a) The task-resource matching subproblem and (b) the task execution priority subproblem.

best experience on the velocity of iteration. r_1 and r_2 are the random numbers between 0 and 1, adopted to increase the diversity of particle movement and prevent particles from moving only toward the individual best experience or the global best experience.

In this work, a constriction PSO is applied to solve the task execution priority problem. Similar to the inertia weight, a constriction factor χ was introduced into PSO to balance global and local searches, i.e., the constriction factor used to limit the movement size, and is named constriction PSO [18]. The velocity update rule used in constriction PSO is indicated in Eq. (5):

$$\begin{cases} V_{ij}^{new} = \chi\left[V_{ij} + c_1 r_1 (gbest - X_{ij}) + c_2 r_2 \left(pbest_{ij} - X_{ij}\right)\right] \\ X_{ij}^{new} = X_{ij} + V_{ij}^{new} \end{cases} \tag{5}$$

The discrete PSO (DPSO) was proposed by Kennedy and Ebrhart in 1997 [19]. Unlike in conventional PSO, the particle position components of the discrete PSO are binary. In other words, $X_{ij} \in (0,1)$. The velocity update is similar to Eq. (5), but the constriction factor is set to 1. The position vector X_{ij} is calculated with the sigmoid function $S(V_{ij})$ in conjunction with velocity vector V_{ij}, and the real valued velocity is converted into a probability. This probability is then compared with a random number to update the position component value either 0 or 1. The position update rule for discrete PSO is displayed in Eq. (6):

$$X_{ij}^{new} = \begin{cases} 0, & S\left(V_{ij}^{new}\right) < \text{rand} \\ 1, & S\left(V_{ij}^{new}\right) \geq \text{rand} \end{cases} \tag{6}$$

The sigmoid function is applied to convert the velocity value into a probability between 0 and 1. However, to prevent the value of $S(V_{ij})$ from becoming too close to 0 or 1, the value of V_{ij} is normally limited to between [$-V_{max}$ and V_{max}]. In this study, DPSO is applied to solve the task-resource matching problem. The particle positions are designed as a two-dimensional matrix $M \times N$. M represents the number of tasks, and N is the number of binary bits, which is the floor function of $\log_2$(resource quantity) plus 1. If the number of resources is R, $N = \lfloor \log_2(R) \rfloor + 1$. In other words, $X_i = [X_{ipq}]$, $p = 1 \sim M$, and $q = 1 \sim N$. The combination of number (N) of binary numbers (after the binary values are converted to metric values) of the p row is the resource allocated for the task p. Suppose that the number of resources is 3 (R = 3), then two bits ($N = \lfloor \log_2(3) \rfloor + 1 = 2$) are required.

3.2. Stochastic greedy in ant colony optimization (ACO)

Stochastic greedy is greedy by chance; it has been well applied in constructing the path of ants in ant colony optimization. Ant colony optimization, which was initially introduced by Dorigo et al. in 1996 [20], imitates the foraging behavior of ants. The ACO was first applied to solve the traveling salesman problem [16, 21]. In ACO, ants lay down pheromones on the foraging paths. The deposited pheromone is the stigmergy used to communicate with ants. The amount of pheromone deposited on a particular foraging path increases with the number of ants traveling along the path. An ant foraging path corresponds to a feasible solution to the studied

scheduling problem; an ant establishes a path by using a transition rule to choose nodes to visit, i.e., each movement is determined by stochastic greedy rule as displayed in Eq. (7):

$$j = \begin{cases} \arg\max\limits_{l \in J_k(i)} \left\{ [\tau(i,l)]^{\alpha} [\eta(i,l)]^{\beta} \right\}, & q \le q_0 \\ J, & q > q_0 \end{cases} \tag{7}$$

where $\tau(i, l)$ denotes the pheromone left on edge (i, l) and $\eta(i, l)$ is the heuristic value. α and β are used to determine the relative importance between the pheromone and the heuristic value. $J_k(i)$ represents the set of neighborhood nodes, which can be visited at node V_i by ant k. And, q_0 is a predefined probability usually set to a higher value, and q is a random number between 0 and 1. The next node j to be visited is chosen from $J_k(i)$. When $q \le q_0$, the node with the highest pheromone times heuristic value is selected as the next node V_j (exploitation). If $q > q_0$, V_j is usually determined from $J_k(i)$ by the roulette wheel selection rule. Restated, an ant has a q_0 probability to visit the node with the highest pheromone times the heuristic value and has a $(1 - q_0)$ probability to visit a node other than the node with the highest pheromone times the heuristic value (exploration).

3.3. Stochastic greedy PSO (SGPSO)

This section will introduce the proposed PSO using a new velocity update rule in this study and the design philosophy behind it. In PSO, the particles always move and search for optimums in the solution space in accordance with the best individual experience and the global best experience. In this work, the global best experience in PSO is similar to the path with the highest pheromone level in ACO. Restated, $gbest_j$ in PSO is regarded as the $max\{[\tau(i,j)]^{\alpha} [\eta(i,j)]^{\beta}\}$ in ACO. Without other guidance, PSO can lead other particles to move toward the current global best experience. As a result, the particles can achieve local optimums at an early iteration and become trapped there (local optimum). This indicates that PSO may converge quickly (premature convergence), but trapping on local optimums is likely to happen. The same situation occurs with ACO, if the ants always choose the path with the highest pheromone times the heuristic value (exploitation), it can also lead to the path of local optimum. Hence, ACO retains a certain probability and uses the roulette wheel selection rule (Eq. (7)) to allow ants to explore paths other than that with the highest pheromone so as to prevent from trapping on local optimum. In ACO, a default parameter *SGP* is adopted to determine the relative importance of exploitation and exploration. When the *SGP* value is large, exploitation of the global best path tends to occur, whereas small *SGP* values are more likely to result in individual exploration. To be with the adequate diversity during search, it is important for PSO to find the optimal solution. Hence, to strengthen the exploration capacity of PSO in the solution search process to keep diversity and avoid getting trapped on local optimums, the search mechanism of ACO based on the stochastic greedy rule was implemented in PSO, and hence a new velocity update rule was designed to replace the velocity update rule of referring to both experiences in Eq. (5). Thus, as a stochastic greedy probability (SGP) for global search, *SGP* is designed. Meanwhile, the roulette wheel selection rule used in ACO is modified, i.e., a particle references its own experience rather than using roulette wheel selection rule to reference other particle's experiences. The design of the proposed velocity update rule is as shown in Eq. (8):

$$
\begin{aligned}
V_{ij}^{new} &= \chi \times \left[V_{ij} + c_1 \times r_1 \times \left(Guidance_{ij} - X_{ij}\right)\right] \\
X_{ij}^{new} &= X_{ij} + V_{ij}^{new} \\
Guidance_{ij} &= \begin{cases} gbest_j, & q \le SGP \ (\text{exploitation}) \\ pbest_{ij}, & q > SGP \ (\text{exploration}) \end{cases}
\end{aligned} \tag{8}
$$

where q is a random number between 0 and 1. When q is larger than SGP, the particle velocity is updated in accordance with the individual best experience ($pbest_{ij}$); when it is smaller than SGP, the particle velocity is updated in accordance with the global best experience ($gbest_j$). Restated, a particle has an SGP probability to search following the global experience (exploitation search) and a (1-SGP) probability to search according to individual experience distributed in solution space (exploration search). Restated, the search behavior of particles is driven by the stochastic greedy rule. The PSO using this new velocity update rule is named stochastic greedy PSO (SGPSO) herein. This SGPSO is applied only in the constriction PSO (Eq. (8)) to solve the task execution priority problem, not in the DPSO that deals with the task-resource

Initialization phase:
- Initializes SGPSO: XSGPSO and VSGPSO and DPSO: XDPSO and VDPSO
- Calculates fitness(X), X=XP-SGPSO+XP-DPSO
- Determines individual experience XP=XP-SGPSO+XP-DPSO
- Determines global experience G=GSGPSO+XDPSO

PSO update phase: including priority determination and resource matching

SGPSO:
- Updates VSGPSO and XSGPSO (Eq. (8))
- Determines tasks priorities Pr
- Repairs Pr to Pr' if violating precedence constraint

DPSO:
- Updates VDPSO (Eq. (4)) and XDPSO (Eq. (6))
- Determines task-resource matching Tr

Schedule generation and fitness calculation phase:
- Generates schedule S based on Pr/Pr' and Tr
- Calculates fitness(Xnew), Xnew=XSGPSO+XDPSO

Experiences update phase:

SGPSO:
- Update individual experience if fitness(Xnew)<fitness(XP)
  XP-SGPSO=XSGPSO
- Update global experience if fitness(G)<fitness(XP)
  GSGPSO=XSGPSO

DPSO:
- Update individual experience if fitness(Xnew)<fitness(XP)
  XP-DPSO=XDPSO
- Update global experience if fitness(G)<fitness(XP)
  GDPSO=XDPSO

Back to PSO update phase when the termination condition not met

Figure 5. Operation procedures of the proposed scheme.

matching problem. The operation procedures of the proposed scheme of applying SGPSO and DPSO to solve the interested workflow application scheduling problem in the grid are listed in **Figure 5**.

3.4. The diversity of SGPSO

The diversity of the swarm during moving in the solution space impacts the solution quality. High diversity of the particles in the initial stage is desired for possible most solution space to find a good seed of search. Conversely, in the latter stage, the particles ought to proceed fine search for the better solution, i.e., low diversity of the population should be provided. To analyze the diversity of the SGPSO, the diversity (*DIV*) of a particle swarm is defined by the average absolute distance of whole particles as given in Eq. (9) [22]:

$$DIV = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{Dis(X_i, X_j)}{C_2^N}$$
$$Dis(X_i, X_j) = \sum_{k=1}^{D} |X_{ik}, X_{jk}| \tag{9}$$

where $Dis(X_i, X_j)$ is the absolute distance between particles X_i and X_j. D is the dimension of the particle, and N is the number of particles.

4. Experimental results and discussion

Since there is not any specific library providing grid task scheduling problems for workflow applications, workflow scheduling problems involving larger numbers of tasks were also designed for this study to test whether the proposed method can also perform well on large-scale problems. Intrinsically, the workflow scheduling problem can be regarded as a derivative of the multimode resource-constrained project scheduling problem (MRCPSP). Therefore, the task precedence constraints of the designed workflow scheduling problems in this work are generated based on the problem cases (J10, J12, J14, J16, J18, J20, and J30) of the MRCPSP in the PSPLIB library; each problem case has 50 different instances generated. The workload of an activity is a randomly produced number with 1000 as the base unit. The data transmission between predecessors and successors is created by using random numbers with 10 as the base unit. The processing ability and the external bandwidth of computing resources in the grid were generated as those in the problem designed by [15] as listed in **Table 1**. **Table 2** illustrates the workflow application example of a J14 instance including workload, number of successors, precedence (successor), and communication cost. In **Table 2**, the tasks 0 and 15 are pseudo tasks representing the start and end. The corresponding DAG of the workflow application instance is displayed in **Figure 6**. The settings of the parameters in constriction and discrete PSOs are $\chi = 0.72984$, $\chi = 1$, and $c_1 = c_2 = 2$. The values of different SGP parameters SGP = {0, 0.1, 0.2, …, 0.9, 1} were tested on the all designed problems (J10, J12, J14, J16, J18, J20, and J30) to understand their influence on the SGPSO performance. To evaluate the performance of the workflow application scheduling on the grid, Avg.Dev(%) is used and defined as in Eq. (10):

$$Dev(\%) = \sum_{i \in \text{instances}} \left(\frac{FIT_i - best_i}{best_i} \times 100\% \right) / |\text{instances}|$$
$$Avg.Dev(\%) = \frac{\sum_{t=1 \sim T} Dev(\%)}{T} \tag{10}$$

Resources	MIPS	Bandwidth
R_1	450	8
R_2	1000	2
R_3	650	10
R_4	1500	8
R_5	800	10
R_6	4000	2
R_7	2000	15
R_8	1250	6
R_9	250	20
R_{10}	750	5

Table 1. Grid environment example.

Task No.	Workload	No. of successors	Successors	Communication cost
0	0	3	1 2 3	0 0 0
1	35,000	2	4 5	14 12
2	7000	2	5 9	5 3
3	5000	2	9 13	7 2
4	28,000	3	6 7 11	11 17 6
5	20,000	3	8 11 14	13 11 13
6	18,000	3	9 10 14	11 16 13
7	4000	2	8 14	7 13
8	27,000	2	10 12	3 3
9	4000	1	12	17
10	19,000	1	13	3
11	32,000	2	12 13	4 4
12	31,000	1	15	13
13	16,000	1	15	11
14	22,000	1	15	3
15	0	0		

Table 2. A workflow application example of J14.

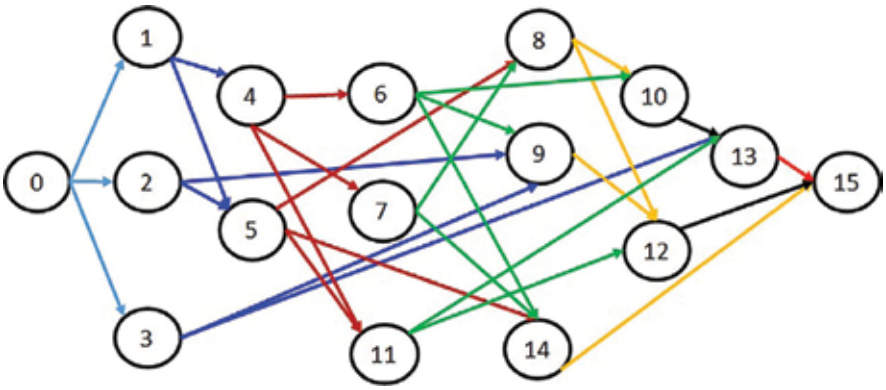

Figure 6. DAG of J14 example in **Table 2**.

where FIT_i indicates the fitness of instance i and $best_i$ is the best known solution of instance i. The |instances| denotes the number of instances of a problem case; it is 50 in this study. Hence, the *Dev*(%) represents the average deviation from the best known solutions which is the best solution found so far. The Avg.Dev(%) is the average of T trials; in this work, 10 trials (T = 10) were conducted.

The performance evaluation on J10 to J30 associates with different SGP values is displayed in **Table 3**. When the SGPs are 0 and 1, the averages of all problem cases' Avg.Dev are 12.43 and 12.05%; they are higher than other Avg.Dev results via other SGPs. This is because only global experience is referred to when SGP = 1 (exploitation only), and convergence of the algorithm in the process of searching in the vast solution space is premature; it is easy to get trapped on local optimums and impossible to obtain the global optimum. When SGP = 0 (exploration

SGP	J10	J12	J14	J16	J18	J20	J30	Avg.
0	3.86	6.61	7.87	15.55	13.11	16.28	23.70	12.43
0.1	3.88	6.58	7.32	15.43	12.14	14.31	17.27	10.99
0.2	4.44	6.87	7.67	15.95	12.39	14.03	15.93	11.04
0.3	4.18	6.74	8.11	15.93	12.46	14.62	15.13	11.02
0.4	4.27	7.27	8.63	16.49	12.54	15.25	14.92	11.34
0.5	4.55	7.26	8.36	16.52	12.49	14.73	15.22	11.30
0.6	4.69	7.82	8.68	16.74	12.72	14.88	15.62	11.59
0.7	4.60	7.39	8.55	17.35	13.21	14.92	15.41	11.63
0.8	4.93	7.50	8.68	17.40	13.13	15.40	14.61	11.66
0.9	4.97	7.77	8.94	16.78	13.38	15.63	15.53	11.86
1	4.79	7.66	9.06	17.36	13.63	16.13	15.72	12.05

Table 3. Performance evaluation on J10 to J30.

only), only local experience is referred to, and slow convergence exhibits while searching in a vast solution space, i.e., obtained optimal solution demands much more iterations.

Meanwhile, the obtained minimum Avg.Dev corresponding to the SGP for each problem case is shown in **Table 4**. The SGP of minimum Avg.Dev for J10 is zero, which is because the best solution in small solution space can be found by exploration search only. The SGPs of minimum Avg.Dev for J12 to J18 and J20 are 0.1 and 0.2, respectively. That is, more exploration search is adequate for small- and medium-scale problems. Since J30 is much more complex than J20, the solution space is vast. Thus, the SGP of the minimum *Avg.Dev* for J30 is 0.8 indicating that more exploitation search for large-scale problems is desired. Restated, the small-scale problem requires smaller SGP, and the large-scale problem needs larger SGP to obtain optimal solution.

Moreover, the comparisons between conventional PSO (ω = 0.8) and the proposed SGPSO (χ = 0.72984, SGP = 0.1) are provided in **Table 5**. **Table 5** indicates that SGPSO outperforms conventional PSO; SGPSO obtains the minimum Avg.Dev for most problem cases except the largest-scale problem (J30). The particle of conventional PSO swarm refers both global and individual experiences to move, i.e., involving both exploitation and exploration during solution search. However, the particle of the SGPSO swarm with SGP = 0.1 mostly refers to individual experience to be the moving guidance, i.e., exploration is carried out during solution search. As concluded above, larger SGP is required to obtain the optimal solution for large-scale problems; hence, SGP = 0.1 conducting more exploration would cause slow convergence. Therefore, when SGP = 0.8 is applied for solving J30, the resulting Avg.Dev (14.61%) is lower than that (15.92%) by using the conventional PSO as shown in **Table 5**.

A resulting schedule of the corresponding DAG of the workflow application scheduling instance with 14 tasks (**Figure 6**) is displayed in **Figure 7**. The fitness evolution of the J14 instance with different SGPs is displayed in **Figure 8**.

Additionally, to further realize the search behavior of the proposed scheme, the diversity evolution of the proposed SGPSO is checked. **Figure 9** displays the diversity evolution of a J14 instance with SGP = 0, SGP = 0.1, SGP = 0.8, and SGP = 1. The diversity to be checked in this study is defined as in Eq. (9).

	J10	**J12**	**J14**	**J16**	**J18**	**J20**	**J30**
SGP	0.0	0.1	0.1	0.1	0.1	0.2	0.8
Avg.Dev	3.86	6.58	7.32	15.43	12.14	14.03	14.61

Table 4. Minimum Avg.Dev corresponds to the *SGP*.

	J10	**J12**	**J14**	**J16**	**J18**	**J20**	**J30**	**Avg.**
SGPSO	3.88	6.58	7.32	15.43	12.14	14.31	17.27	10.99
PSO	4.52	7.42	8.72	20.55	13.12	15.12	15.92	12.20

Table 5. Comparisons between conventional PSO and SGPSO (*SGP* = 0.1).

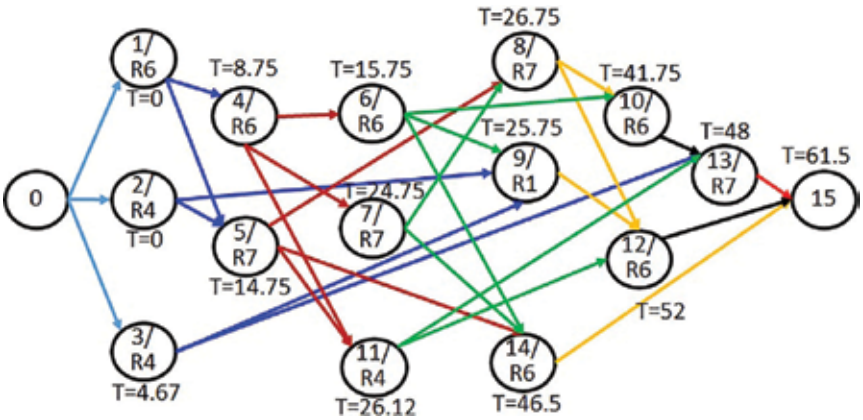

Figure 7. Workflow schedule of the J14 instance.

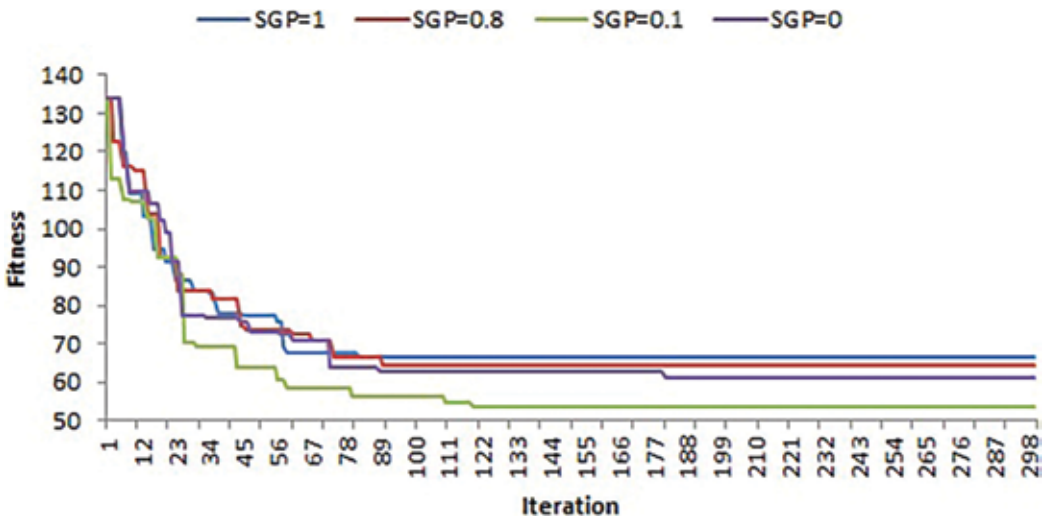

Figure 8. Fitness evolutions on J14 with different SGPs.

In **Figure 9**, the diversity remains high until the end of the operation when SGP = 0 without referencing global experience. Restated, particle search behavior keeps exploration until the end and hence suffers slow convergence. Conversely, the diversity quickly drops to none for SGP = 1, i.e., the particles go to exploitation search and therefore lead to premature convergence and trap on local optima. When SGP = 0.8, the behavior is similar to that of SGP = 1 but provides some exploration ability. Hence, SGP = 0.8 has the higher diversity than that of SGP = 1. With the setting of SGP = 0.1, high diversity in the early stage and diversity gradually lowered after the middle stage to the end are provided. Therefore, giving global search in the early stage gradually shrinks the search area in the later stage, hence providing an ideal search process from exploration to exploitation for finding the optimal schedule.

Finally, this work tested the workflow application problems examined in [15] to verify the proposed method. The comparison is made mainly with the largest-scale case in [15] which involves 15 tasks (represented here as J15).

The settings of the parameters in constriction and discrete PSOs are $\chi = 0.72984$, $\omega = 1$, and $c1 = c2 = 2$. The values of different SGP parameters SGP = {0, 0.1, 0.2, …, 0.9, 1} were tested. The minimum average fitness of the simulation results of iteration = 300 is shown in **Figure 10**.

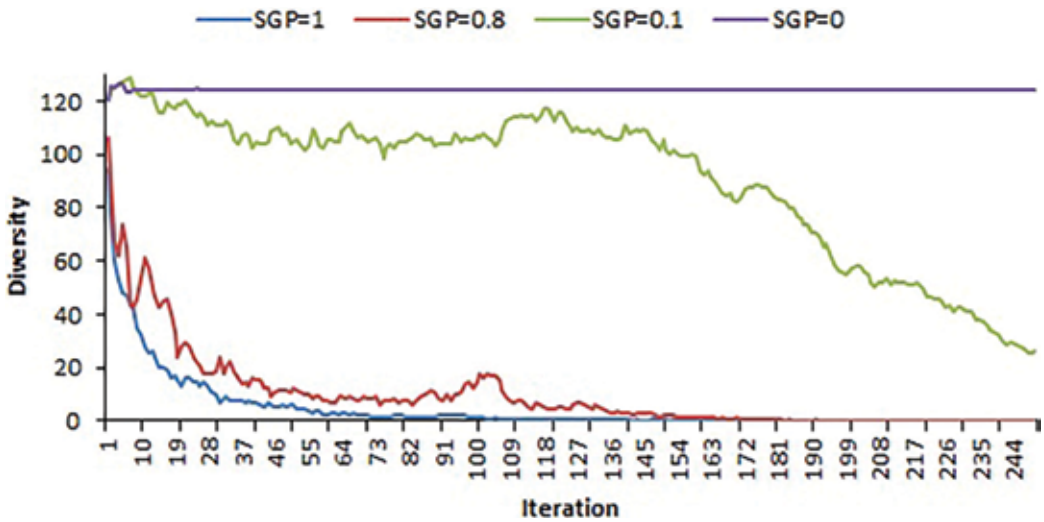

Figure 9. The diversity evolutions of the swarm of a J14 instance with different SGPs.

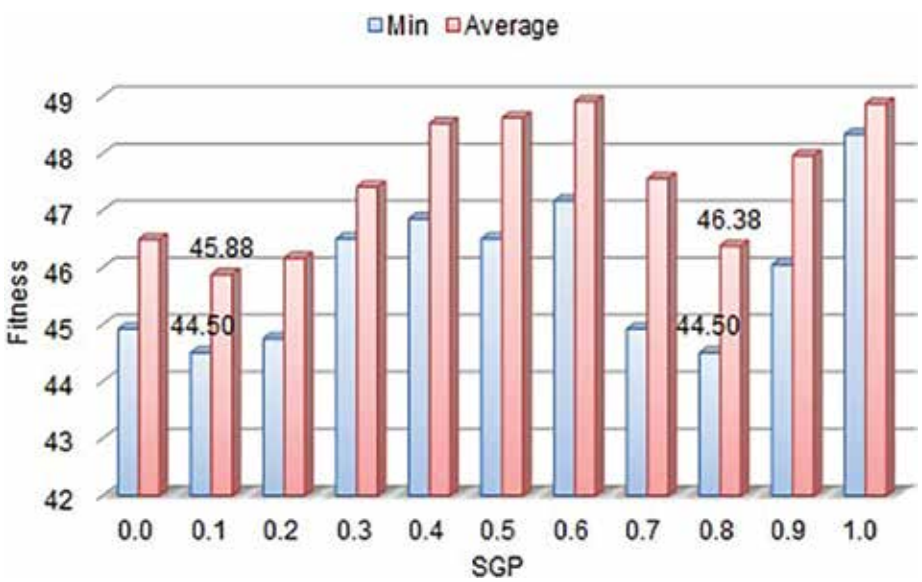

Figure 10. The minimum average fitness of J15.

Figure 10 shows that the minimal fitness (44.5) can be obtained when SGP = 0.1 and SGP = 0.8. However, the minimum average fitness (45.88) is obtained when SGP = 0.1. The results coincide with the above experiment consequences. The comparison between this work and [15] is listed in **Table 6**.

According to **Table 6**, with a small SGP value, the task scheduling outcomes of J15 will become better as the number of iterations increases (100 iterations → 300 iterations). This is because small SGP values tend to lead to the use of individual experience. In other words, the particles perform exploration search in solution space. In consequence, to obtain the optimums in the vast solution space will consume much time, and the convergence is delayed.

		Chen [15]	**Chen [15]**	**This work**
		χ = 0.75	**χ = 0.5**	**χ = 0.72984, SGP = 0.1**
100 iter.	Min.	45.50	44.50	46.75
	Avg.	60.15	54.26	47.71
300 iter.	Min.	44.50	44.50	44.50
	Avg.	55.45	50.36	45.88

Table 6. Performance comparison on J15 in [15].

5. Conclusions

Workflow application is the most common application in the grid. However, the workflow scheduling heavily affects the performance of workflow execution application. Two PSOs were used to solve task-resource matching and task execution priority subproblems of the workflow scheduling. A new and simplified velocity update rule extended from the ACO state transition rule is designed in constriction PSO for solving the task execution priority subproblem. Restated, the search control is based on a suggested SGP inspired by the ACO's transition rule. This constriction PSO-based algorithm is named stochastic greedy PSO (SGPSO), which provides both exploration and exploitation abilities during search. The main purpose is to strengthen the exploration capacity of the PSO in the solution search process while providing certain exploitation capability to avoid getting trapped in local optimums.

According to experimental results as indicated in **Table 3**, high SGP provides global experience guidance and causes premature convergence, hence easy to trap on local optimal such as only exploitation applied in SGP = 1.0 and Avg.Dev = 12.05% yielded. When SGP = 0, the algorithm would conduct self-search such that only exploration is enabled that causes slow convergence and Avg.Dev = 12.43% obtained. Better solutions can be found while providing enough exploration and certain exploitation capabilities such as SGP = 0.1; the lowest Avg. Dev = 10.99% can be obtained.

By using the SGP to control the search behavior, either exploitation or exploration would make the algorithm simplified and also reduce the execution time.

Meanwhile, high diversity in the early stage and low diversity in the later stage are preferred for searching in the solution space provided as indicated in **Figure 9**. Therefore, the proposed SGPSO with lower SGP is suggested for solving workflow class scheduling problem in the grid.

Unlike in [15], using heuristics to find initial solutions is not adopted in this work. Therefore, there is no need to consider which heuristic should be designed to increase performance, and hence the algorithm is thus easier to implement. In [15], the best result comes with the constriction factor $\chi = 0.5$ which was obtained after thorough testing. However, the best result can be yielded with the commonly suggested value $\chi = 0.72984$. Hence, the laborious work of finding the best constriction factor value is eliminated.

The experimental results show that the proposed method can effectively solve grid task scheduling problems and boost grid performance. In reality, there are many problems similar to grid task scheduling problems, such as the multimode resource-constrained project scheduling problems (MRCPSPs). In the future, the method proposed in this study will be applied to find solutions to MRCPSP-type problems.

Acknowledgements

This work was partly funded and supported by the Ministry of Science and Technology, Taiwan, under contract MOST 104-2221-E-167-011.

Conflicts of interest

The authors declare no conflict of interest.

Author details

Ruey-Maw Chen[1]* and Yin-Mou Shen[2]

*Address all correspondence to: rmchen@mail.ncut.edu.tw

1 Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taiwan

2 Department of Information Management, Kun Shan University, Taiwan

References

[1] Beynon M, Sussman A, Catalyurek U, Kurc T, Saltz J. Performance optimization for data intensive grid applications. In: Proceedings of the Third Annual International Workshop on Active Middleware Services, San Francisco, CA, USA, Aug 2001, AMS'01. Los Alamitos, USA: IEEE Computer Society Press

[2] Goux JP, Kulkarni S, Linderoth J, Yoder M. An enabling framework for master-worker applications on the computational grid. In: Proceedings of The Ninth International Symposium on High-Performance Distributed Computing, Pittsburgh, USA, Aug 1–4, 2000. Los Alamitos, USA: IEEE Computer Society Press

[3] Xhafa F, Paniagua C, Barolli L, Caballé S. A parallel grid-based implementation for real time processing of event log data in collaborative applications. International Journal of Web and Grid Services. 2010;**6**(2):124-140

[4] Chang RS, Lin CY, Lin CF. An adaptive scoring job scheduling algorithm for grid computing. Information Sciences. 2012;**207**:79-89

[5] Fidanova, S. Simulated annealing for grid scheduling problem. In: Proceedings of the IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing (JVA'06), Sofia, Bulgaria, Oct 3–6, 2006. Los Alamitos, USA: IEEE Computer Society Press

[6] Gao Y, Rong H, Huang JZ. Adaptive grid job scheduling with genetic algorithms. Future Generation Computer Systems. 2005;**21**:151-161

[7] Liu H, Abraham A, Hassanien AE. Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. Future Generation Computer Systems. 2010; **26**(8):1336-1343

[8] Lorpunmanee S, Sap MN, Adbullah AH, Chai C. An Ant Colony Optimization for dynamic job scheduling in grid environment. Engineering and Technology. 2007;**1**(5): 314-321

[9] Salimi R, Motameni H, Omranpour H. Task scheduling using NSGA II with fuzzy adaptive operators for computational grids. Journal of Parallel and Distributed Computing. 2014;**74**(5):2333-2350

[10] Chen RM, Shen YM, Wang CT. Ant Colony Optimization inspired swarm optimization for grid task scheduling. In: Proceedings of the 2016 International Symposium on Computer, Consumer and Control (IS3C 2016), Xi'an, China, July 4–6, 2016. Los Alamitos, USA: IEEE Computer Society Press

[11] Chen RM, Shih HF. Solving university course timetabling problems using constriction particle swarm optimization with local search. Algorithms. 2013;**6**:227-244

[12] Li D, Deng N. Solving permutation flow shop scheduling problem with a cooperative multi-swarm PSO algorithm. Journal of Information & Computational Science. 2012;**9**(4): 977-987

[13] Chen RM, Shen YM. Novel encoding and routing balance insertion based particle swarm optimization with application to optimal CVRP depot location determination. Mathematical Problems in Engineering. 2015;**2015**:11

[14] Tao Q, Chang HY, Yi Y, Gu CQ, Li WJ. A rotary chaotic PSO algorithm for trustworthy scheduling of a grid work flow. Computers & Operations Research. 2010;**38**(5):824-836

[15] Chen RM, Wang CM. Project scheduling heuristics-based standard PSO for task-resource assignment in heterogeneous grid. Abstract and Applied Analysis. 2011;**2011**:20

[16] Dorigo M, Gambardella LM. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation. 1997;**1**(1): 53-66

[17] Kennedy J, Ebrhart RC. Particle swarm optimization. In: Proceedings of the 4th IEEE International Conference on Neural Networks, 1995, Perth, Australia, Nov 27–Dec 1, 1995. Piscataway, NJ: IEEE Service Center; 1995

[18] Bratton D, Kennedy J. Defining a standard for particle swarm optimization. In: Proceedings of the IEEE Swarm Intelligence Symposium (SIS'07), Honolulu, UAS, Apr 1–5, 2007. Los Alamitos, USA: IEEE Computer Society Press

[19] Kennedy J, Ebrhart RC. A discrete binary version of the particle swarm Algorithm. In: Proceedings of The IEEE International Conference on Systems, Man, and Cybernetics, Orlando, USA, Oct 12–15, 1997. Vol. 5. Los Alamitos, USA: IEEE Computer Society Press

[20] Dorigo M, Maniezzo V, Colorni A. The ant system: Optimization by a colony of cooperating agents. IEEE transaction on system. Man and Cybernetics. 1996;**26**(1):1-13

[21] Hlaing ZCSU, Khine MA. Solving traveling salesman problem by using improved ant Colony optimization algorithm. International Journal of Information and Education Technology. 2011;**1**(5):404-409

[22] Chen D, Wang J, Zou F, Hou W, Zhao C. An improved group search optimizer with operation of quantum-behaved swarm and its application. Applied Soft Computing. 2012;**12**(2):712-725

Chapter 5

Performance Comparison of PSO and Its New Variants in the Context of VLSI Global Routing

Subhrapratim Nath, Jamuna Kanta Sing and
Subir Kumar Sarkar

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.72811

Abstract

Substantial reduction of gate delay occurred in recent times owing to radical decrement of transistor size. The interconnect length and delay are accordingly increased owing to the exponential escalation of packaging density with additional transistors being fabricated on the same chip area. The function of VLSI routing that seems to be more defying to the scholars, is categorized in global routing and detailed routing phase. In global routing phase, the prevalent method to lessen the wire length for reducing interconnect delay is to adjust the cost of the Steiner tree, devised by the terminal nodes to be interconnected. Nevertheless, Steiner tree problem is a NP-complete problem in classical graph theory where meta-heuristics might impart beneficial elucidations. Particle swarm optimization (PSO) is a robust algorithm concerning VLSI routing field. This chapter is regarding the proposal of a self-adaptive mechanism for monitoring acceleration coefficient of PSO and evaluating its functionalities with the existing acceleration coefficient controlled PSO in numerous allocation topologies of terminal nodes within definite VLSI layout. The outcomes of PSO variant with constriction factor in context to VLSI route reduction ability and robustness are also inspected. Additionally, a new effort in adapting the PSO with embracement of genetic algorithm is established.

Keywords: VLSI, global routing, Steiner tree, meta heuristics, PSO

1. Introduction

Stating optimization as a usual phenomenon is a bit of exaggeration, which includes economic development to engineering strategy as well as job scheduling to resource allocation. The intention of optimization is certainly to produce comparable outcomes under specified conditions bearing some parametric minimization or maximization. In VLSI physical design context

numerous parameters are needed to be optimized like transistor count, power delay product, interconnect delay.

Conferring to Moore's Law, the transistor count doubles up in every 18 months [1]. Since the dimensions of the transistors are radically diminishing by the contemporary ages as a result of massive development in technologies, additional transistors are getting assimilated in that single chip region than before by means of cutting-edge assembling techniques. Therefore the length of interconnect has also been considerably amplified. Previously it was sufficient to overhaul the gate delay but interconnect delay has been more noticeable after the 130 nm technological node was pioneered.

The objective of VLSI physical design is to optimize the devices arrangements and interconnection schemes among these devices for desired performance.

Wire-length approximation of interconnects is considered in the routing phase of VLSI physical design process, which is largely categorized into Global Routing and Detailed Routing. In Global routing phase the circuit interconnections in shortest possible wavelength and minimum interconnect delay is required to be achieved. The complexities of global routing problem is solved to some extent with sequential approach where VLSI nets are sequenced according to their criticality and practical routers employs improvement phase. Technique of rerouting after ripping interfering wires [2] and 'shove-aside' technique [3] and also introducing concurrent approach where parallel integer programming concept are tried for enhancement of global routing but with limitations.

The Routing problem of VLSI physical design can also be mapped in classical Graph Theory where wire-length minimization of interconnected nodes rests in solving the Rectilinear Minimal Steiner Tree Problem (RMST) [4], a renowned NP Complete problem of Graph Theory. Such NP complete problems can be aimed to solve by a division of Artificial Intelligence known as Swarm Intelligence. Swarms interact among themselves to persist in any situation. It has been observed that these social agents have restricted competences of their own as an individual, however as an assemblage they are capable of accomplishing an assignment, somewhat perceptively for their existence. Scientists and engineers were ardent to mimic these activities of these natural swarm systems. Swarm intelligence was maidenly commenced in 1989 by G Beni and J Wang [5] to crack some practical problems associated to global optimization. These algorithms are heuristics and meta-heuristics in character. Heuristic infers "to ascertain by trial and error". These approaches are fairly beneficial in obtaining optimal solution or near optimal solution aimed at a complex optimization problem (like NP-complete problems) within a judicious time frame. Meta-heuristics ("meta" means "beyond" or "higher level") conversely execute even better than heuristic algorithms herein because they comprise of precise compromises related to randomization & local search. One such prevalent meta-heuristics algorithm is PSO (Particle Swarm Optimization) [6].

In VLSI system for solving the Routing problem the researcher Dong et al. [7] used PSO in 2009. The proposal by the authors was mainly novel encoding and updating schemes for a discrete or binary version of PSO where Interconnect delay is one of the most important disadvantage. A routing scheme based on PSO combined with buffer insertion at suitable intervals, was taken into consideration for overcoming the disadvantage, as proposed by Ayob

et al. [8]. Again a PSO algorithm was proposed by Liu et al. [9] to reduce binding during routing where this version of PSO algorithm was successful in reducing the binding problem but in turn it increased the cost of the wire lengths. Among many other proposed algorithm, the one proposed by Shen et al. [10] has some important significance which dealt with the self-adaptive technique of inertia weight update. Many other technique based on SI has been used for escalation of VLSI routing, among them the one proposed by Arora and Moses is important. Both Manhattan as well as a non- Manhattan routing scheme based on Ant Colony Optimization (ACO) [11] were proposed by the duo. A proposal was introduced by Ayob et al to evade VLSI routing scheme by using Firefly optimization [12].

Two algorithms centered on inertia weighted PSO (PSO-W) [13] are presented in here. In the prior one, named as Self-Adaptive acceleration coefficient PSO (PSO-SAAC), the acceleration coefficients are adjusted by the means of an adaptive procedure of local search and global search to heighten the property of the searching technique composed with efficient converging rate. Additional new tactic is implemented where the perception of the genetic algorithm is hybridized with PSO-W by comprising a component related with a breading factor in the position update characteristic equation of PSO-W. In supplement to the above two alternatives of PSO, PSO with constriction factor (PSO-C) [14] are evaluated with these algorithms, most lately familiarized, on a mutual platform of optimization of VLSI global routing. Further all the algorithms are verified in several distribution topologies of VLSI terminal nodes in a definite search area.

The remaining chapter is arranged accordingly. Section 2 portrays the elementary theory of VLSI Global routing, RMST and PSO. In Section 3 algorithms variants on PSO-W are illustrated in specifics shadowed by the implementation of modified PSO algorithms in Section 4. Section 5 confers the experimental results acquired in context to VSLI global routing and lastly the chapter gets concluded with Section 6.

2. Preliminaries

2.1. Global routing in VLSI physical design

VLSI routing in Physical Design context initiates with the procedure of interconnections amongst the circuit blocks and pins, specified according to the net list, generate results in the phase of placement. The inputs to the general routing problem are as follows:

- Net list.
- Timing budget for the critical nets.
- RC delay of per unit length of metal layers and vias.

Conventionally, the routing fashion can be broadly cleft into two main stages: the initial stage, also called as the “Global Routing” allocates a list of routing areas for individual net, putting aside the absolute geometrical blueprint of wires; whereas, the secondary stage, called as “Detailed Routing”, finds the absolute geometrical blueprint of any net within the allocated routing areas.

The purpose of the routing problem is to curtail the wire length, at the same time accommodating the timing budget for the critical nets. Global routing is the initial phase of routing

where a catalogue of routing regions are essentially allocated for a net, in default of stipulating the tangible geometric layout of interconnects, acknowledging that no net could be crisscrossing each other.

Now, the complications related to global routing can be overcome using two basic approaches-the sequential approach and the concurrent approach.

The term 'sequential' in sequential approach refers to something which occurs in a sequence of steps; and here as the name indicates, nets are connected in succession. This approach of routing is very susceptible to the order in which the nets are considered for routing, because a net once routed may hinder other nets. The parameters based on which the nets are ordered includes half of the total area of the confining polygon, their criticality, and the number of terminals. High criticality number is assigned to the net on the critical path as the performance of the circuit is dependent on them to a great extent. But nonetheless these techniques of sequencing are not impeccable. As a blunt aftermath of this, a factual router engages a development or improvement phase besides the sequencing phase to do away with the jam when no more routing is possible. But still this may not conquer the frailty of the sequential approach. 'Rip-up and reroute' [2] and 'shove-aside' techniques [3] are examples of improvement phase like this. As the name indicates, in case of the 'rip-up and reroute', the intrusive cables are ripped up and rerouted; but in case of the 'shove aside' technique, cables which make is viable to outright the failed connections are put aside without curbing the extant connections in any way.

In Concurrent approach, by simultaneously considering all the nets, the complications related to ordering can be evaded. There is no compelling polynomial algorithm (not even for nets with only two terminals) and as a consequence this approach is computationally tougher. Ergo, it was proposed to use integer programming; nonetheless as a consequence of the very large size of the resulting program, it can't be utilized efficiently. So, to overcome this problem, the program is fragmented into smaller sub programs using hierarchical method (which follows top down approach), which can be then readily dealt with by integer programming. A sequence of routing channels is accredited to each net by global routing without contravening the capacity of channels. At the same time, the total length of the wire is also occasionally revamped.

The intention of the routing problem is reliant on the attributes of IC which is to be fabricated. Nowadays, VLSI chips contain up to billions of transistors which makes it possible to complete a layout to route millions of nets. In turn there may be several thousands of routes in each net. It comprises the trade-off between routability of all nets and minimization of the wire length in interconnects. The objective function of this routing puzzle is an eminent NP-complete problem. All these attributes results the routing problem to be a computationally tough one which furnishes the scope for metaheuristic algorithm, like for resolving the problem, SI based algorithm are to be used.

2.2. Rectilinear Steiner tree problem

Rectilinear Steiner Tree (RST) is the resulting of Minimum Spanning Tree (MST) for a specified grid graph. For a certain set of vertices of a graph, minimum spanning tree is shaped by interconnecting them where MST is the maximal sub graph and the MST cost is the sum of the weights of all edges in the tree. Certain additional intermediate vertices are appended with MST

to create Steiner tree in turn to lessen the entire length of the MST where intermediate vertices are Steiner points. The challenges remain in selecting the number and position of Steiner vertices in the Gird layout. Steiner tree with restricted edge to be in rectilinear in rectangular grid graph originates RST. This tool is utilized to acquire the least possible length of the inter connections for a specified set of nodes in the grid graph. RST, although NP complete problem [15], is an efficient method in improving the length of interconnects in VLSI circuits.

Nonetheless some other constraints for instance noise, power, electro-migration, signal integrity, packaging density, skew, inductance, reliability etc., frequently have vast impact on the objective function in deep-submicron VLSI design, the length of the non-critical nets however preserves its significance in wire length minimization.

2.3. Particle swarm optimization

PSO is a multi-agent equivalent search technique which engages incorporates an iterative method to obtain the ideal solution in a multi-dimensional search space. Assume that there exists a d dimensional search space, where the number of agents arbitrarily allocated are n. The agents are primed with certain position and velocity vectors as $X_i = \{X_1, X_3, \ldots\ldots, X_n\}$ and $V_i = \{V_1, V_2, \ldots\ldots, V_n\}$ respectively. These vectors are periodically updated rendering to the characteristic equations of PSO as given in Eqs. (1) and (2).

$$V_{i,t+1} = V_{i,t} + c_1 * r_1 * \left(p_{best} - X_{i,t}\right) + c_2 * r_2 \left(g_{best} - X_{i,t}\right) \tag{1}$$

$$X_{i,t+1} = X_{i,t} + V_{i,t+i} \tag{2}$$

Where the constants c_1 and c_2 are accountable for the impact of the distinctive particle's individual information and so as of the group information correspondingly. The variables r_1 and $r_2$are unvaryingly distributed random numbers [16]. All particles are adjusted randomly and keep on promoting the fitness value influenced by the p_{best} value (best position value of the individual) and g_{best} value (best position value of the entire swarm) correspondingly in anticipation of the optimal solution to be accomplished.

2.4. PSO parameters

2.4.1. Swarm size

Swarm size infers the number of particles existing in the swarm. A huge number of particles can exploit a huge extent of a search space; therefore fewer iterations are required so as to achieve the optimal solution. Contrariwise an enormous swarm size upsurges the computational complexity and time complexity likewise.

2.4.2. Iteration number

Number of iterations is a problem contingent parameter related to swarm size. An inadequate number of iterations can terminate the program precipitately earlier to the conjunction whereas huge number of iterations generates a redundant computational and time complexity.

2.4.3. Velocity component

The velocity update factor in Eq. (1) comprises of three terms. The first term is the preceding velocity vector i.e. the former direction & the magnitude of the velocity of a particle. This component averts a particle from a radical alteration in velocity in current iteration. The following component is the cognitive one. It is centered on the memory of an agent in agreement with its proficiency. The cognitive component continuously inspire a particle to reappear to its position, suitable for it in local. The subsequent component is the social component. This is the knowledge to an individual by social communication, which constantly encourage the particle to travel in the direction of the best position, knowledgeable by its vicinity.

2.4.4. Acceleration coefficients

The variables c_1 and c_2 are known as acceleration coefficients, which attempt to generate an equilibrium amid the cognitive component and social component of the velocity.

- If $c_1 = c_2 = 0$, Eq. (1) will be $V_{i,t} + 1 = V_{i,t}$. This implies that all the particles retain to hover with their initial velocity, ensuing no search condition.
- If $c_1 > 0$ and $c_2 = 0$, Eq. (1) resolves to $V_{i,t+1} = V_t + c_1 * r_1 * (p_{best} - X_{i,t}) + 0$. This implies that all the particles revolve around their searching space autonomously. Since they are not interacting with the neighbours, they are incapable to obtain the global optimal solution whatsoever.
- If $c_1 = 0$ and $c_2 > 0$, Eq. (1) resolves to $V_{i,t+1} = V_t + 0 + c_2 * r_2 * (g_{best} - X_{i,t})$. It infers that all particles are fascinated to a single point, which is not revised in each time step.
- If $c_1 = c_2$, all particles will travel towards average p_{best} and g_{best} values.
- If $c_1 \gg c_2$, it results in manipulating the particles in the direction of p_{best} position and $c_2 \gg c_1$, resulting the particles to be enticed towards the g_{best} position and in both circumstances the particles sprint precipitately to the optimum solution.

Usually c_1 and c_2 are considered as equal, constant values and various intellectual articles propose their values as 2 for getting decent optimal results [14].

3. Analysis of PSO characteristics & modification

3.1. Velocity clamping

Particle's velocity, a significant parameter of PSO algorithm, is the step size of swarm in every iteration. With all time step, the particles alter their velocity & travel in all direction in the problem space. If the velocity is extreme, the assessment attribute of the particle turn out to be high and simultaneously the particle might hastily vacant the periphery of the search space and swerve. On a converse if velocity is low, the movement of particles is limited upon a small boundary and it happens confined in a local optima. Therefore, it is required to preserve an equilibrium amidst exploration & exploitation by situating a parameter V_{max}, assumed as $V_{max} = \frac{(X_{max} - X_{min})}{k}$. The empirical value of k is set as 2 [14].

3.2. Inertia weight

Inertia Weight (w) was additionally familiarized [13] progressing PSO-W to substitute V_{max}, to regulate the momentum of the particle in assessing the updated velocity. It is presented to regulate the exploration and exploitation aptitudes of the swarm with the intention of the algorithm to converge more efficiently upon time. Therefore Eq. (1) is adapted as Eq. (3).

$$V_{i,t+1} = w*V_{i,t} + c_1*r_1*\left(p_{best} - X_{i,t}\right) + c_2*r_2\left(g_{best} - X_{i,t}\right) \tag{3}$$

- If w = 1: then Eq. (3) is similar to the original Eq. (1).
- If w > 1: then the velocity will increase over time and the particles will barely be capable of altering their direction.
- If w < 1: particles can rapidly alter their route subjective to pbest and gbest values.
- If w = 0: particles travel lack of any acquaintance of the preceding velocity.

Typically the inertia weight w is selected dependent to the size of the search space. A high value of w is essential for complex high dimensional problem space and trivial value for small dimensional search space.

The inertia weight can be differed by Eq. (4), where *s* is the population size, *D* is the Dimension size and *R* is relative quality of corresponding solution standardized to [0,1].

$$w\left[3 - \exp\left(\frac{-s}{200}\right) + \left(\frac{R}{8}*D\right)^2\right]^{-1} \tag{4}$$

3.3. Constriction factor

The PSO algorithm is reorganized to substitute the inertia weight w & max velocity V_{max} by a fresh parameter χ, known as constriction factor given in Eq. (6). Clerc [17] pioneered this factor, which proved to be exceptionally significant in regulating the exploration & exploitation trade-off, thus guaranteeing an efficient conjunction of algorithm. Eq. (1) gets amended as Eq. (5).

$$V_{i,t+1} = \chi*\left[V_{i,t} + \Phi_1*\left(p_{best} - X_{i,t}\right) + \Phi_2*\left(g_{best} - X_{i,t}\right)\right] \tag{5}$$

$$\chi = \frac{2}{\left[2 - \phi - \sqrt{\phi^2 - 4\phi}\right]} \tag{6}$$

Here, $\phi = \phi_1 + \phi_2$, $\phi_1 = c_1*r_1$ and $\phi_2 = c_2*r_2$. Characteristically applying the value of ϕ as 4.1 the value of χ results to 0.729. Therefore, $\chi*w = 0.729*w < w$, infers that the particles rapidly alter their course manipulated by p_{best} and g_{best} with assured convergence. Both $\left(p_{best} - X_{i,t}\right)$and $\left(g_{best} - X_{i,t}\right)$are multiplied by $2*0.729 = 1.458$ [18]. Generally these values are preferred for improved stability and convergence.

3.4. Acceleration coefficient

Acceleration coefficient as a PSO parameter has previously been explained in the preceding segment. Usually both the values of c_1 and c_2 are applied to be 2 [19]. The equilibrium concerning

these parameters can be monitored in two distinct ways, mentioned underneath, to accomplish superior result in perspective of path minimization of VLSI global routing.

3.4.1. Self-tuned

In this algorithm PSO-ST [20], the acceleration constants both c_1 and c_2 are reduced linearly throughout the time steps in the range of 2 to 1.49. At the beginning, the algorithm is primed with c_1=c_2 =2. By this changing of linear decrement, both exploration and exploitation abilities of the swarm can be preserved efficiently for velocities updating and can deliver a swift convergence to the algorithm. This algorithm turns out to be competent to obtain optimal result with lofty convergence rate.

3.4.2. Self-adaptation

An algorithm PSO-SAAC is introduced where the two acceleration constant parameters c_1 and c_2 have been assorted in such a style that they got enhanced influence over the trade-off in between global exploration and local exploitation. The algorithm commences with highest exploration and lowest exploitation aptitudes of swarm, which have eventually been altered in every time step over the entire iteration process. Therefore the particles of the swarm are capable of dispersing all over the search space consistently, motivated by the social component of the velocity vector at the first phase of experiment. Since the cognitive component outpace the social component in the subsequent phase of the experiment, the swarm accomplish the local search process entered on the assessed results of the Global search process with the intention of obtaining the finest local optima. Throughout the whole searching process this self-adaptive procedure can be effectual in producing most significant g_{best} value and thus in this manner heightening the optimization rate.

3.5. PSO with mutation

A fresh algorithm is presented where the principle of Genetic Algorithm is featured in PSO [21]. The algorithm after utilizing some time steps initiates with selection of swarms from existing generation in the first phase. The swarms with high fitness probability get selected where the probability of selection factor $is\frac{f_j}{\sum_{j=1}^{N} f_j}$, where N is the population size. The high fitness factor is extracted from the selected pool generating a mutant in the second phase. This enhanced knowledge of high fitness property is induced in the position vector Eq. (2) to evolve a new generation of swarms causing mutation in PSO [22]. The proposed position vector in Eq. (7) is given below.

$$X_{i,\,t+1} = (\psi * X_{i,\,t} + \xi) + V_{i,\,t+1} \tag{7}$$

where, ψ is the randomization factor and ξ is the mutant fitness factor.

4. Problem formulation and implementation

The challenge of global routing problem of a VLSI Physical design rests in two objectives. First in minimizing power dissipation and secondly increase the speed of signaling among the

http://dx.doi.org/10.5772/intechopen.72811

partitions or blocks in VLSI layout which can be achieved by reducing the complete wire length of interconnected terminals or blocks. The Global routing problem can be formally stated where $N=\{N_1,N_2,N_3 \ldots N_m\}$ be the set of nets denoting interconnections among blocks in the VLSI layout and the estimated wirelength of net N_i, $1< i < m$ is denoted by D_i. The problem function can be expressed such that overall total wirelength $\sum_{i=1}^{m} D_i$is minimized. Global routing formulation is done by mapping the required VLSI layout in classical graph theory as Grid Graph model. Here the grid graph model is regarded as to execute the above proposed algorithms. The grid graph, $G =$), is an exemplification of a routing region layout where region is carved into a number of unit square cells as shown in **Figure 1**. Each cell representing routing areas between blocks as empty area is signified by vertex a v_i and the edge e_{ij}, linking the two neighboring vertices v_i and v_j. The vertices resemble to the nodes and edges resemble to the routing paths between blocks in a VLSI layout.

To obtain the solution of the VLSI routing problem for a multi-terminal net, the primary assignment is to articulate it as the problem of obtaining an RMST (Rectilinear Minimum Spanning Tree) from a Graph. The Minimum spanning tree of the interconnected terminal nodes is generated using graph algorithms results in measuring the minimum cost of interconnected length. With introduction of random Steiner nodes along with the terminal nodes of multi-terminal VLSI layout the cost or the overall wirelength is further reduced generating the minimum Steiner tree cost (length) in the graph. Depending on the position and the number of Steiner nodes the cost or overall length can be further minimized. With large number of terminal nodes the probability of determining the number of Steiner nodes and desired positioning of these Steiner node become computationally hard and hence the PSO algorithm is used to select probable number of Steiner nodes and generate these random position in order to optimize the Steiner cost.

The algorithm commenced with random generation of swarms size of z particles and they are placed in required graph of $n \times n$ dimension. Each of the swarm consists at most $(p\text{-}2)$ randomly generated Steiner points drawn from Steiner set S with $(n||2\text{–}p)$points where p is

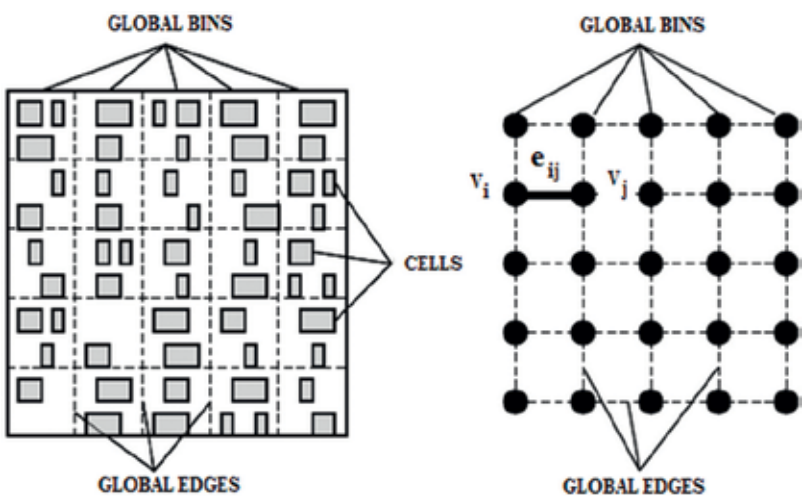

Figure 1. Routing region layout in grid graph.

the number of terminal nodes with designated vertex V_i, $i = \{1,2,3,\ldots r\}$ and thereby Steiner subset $Q^j \subseteq S$, where $j = \{1, 2, 3\ldots\ldots, z\}$ is formed. 100 × 100 search space is used. The defined destination nodes or terminal nodes are represented by 1 to generate the problem matrix. Rows & Columns without the destination nodes are eliminated to reduce computational complexity generating the reduced matrix Steiner points are introduced in a randomly in the problem space, is denoted by 1. The reduced matrix and the corresponding Steiner matrix, are shown in **Figure 2**.

For implementation of PSO, mapping is done with the creation of swarm particles where each these Steiner matrix is considered as a particle. One such particle with the destination nodes is shown is **Figure 3**. Fitness F_i for each particle seed is calculated by evaluating Minimum Rectilinear Steiner Tree (MRST) cost using objective function $MST(G_i)$ and also $MIN\ (MST(G_i))$ which

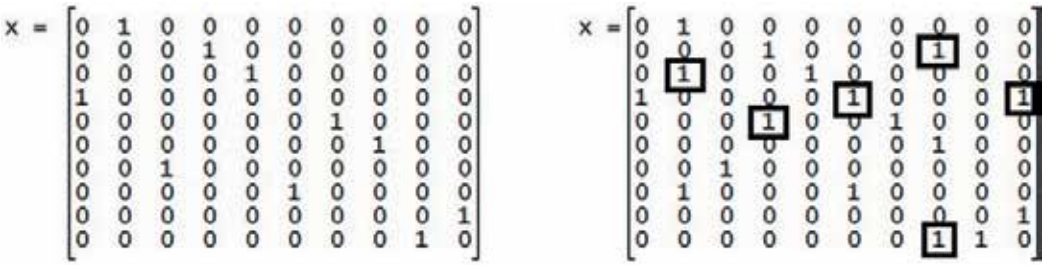

Figure 2. Matrix generated from reduced graph and corresponding Steiner matrix.

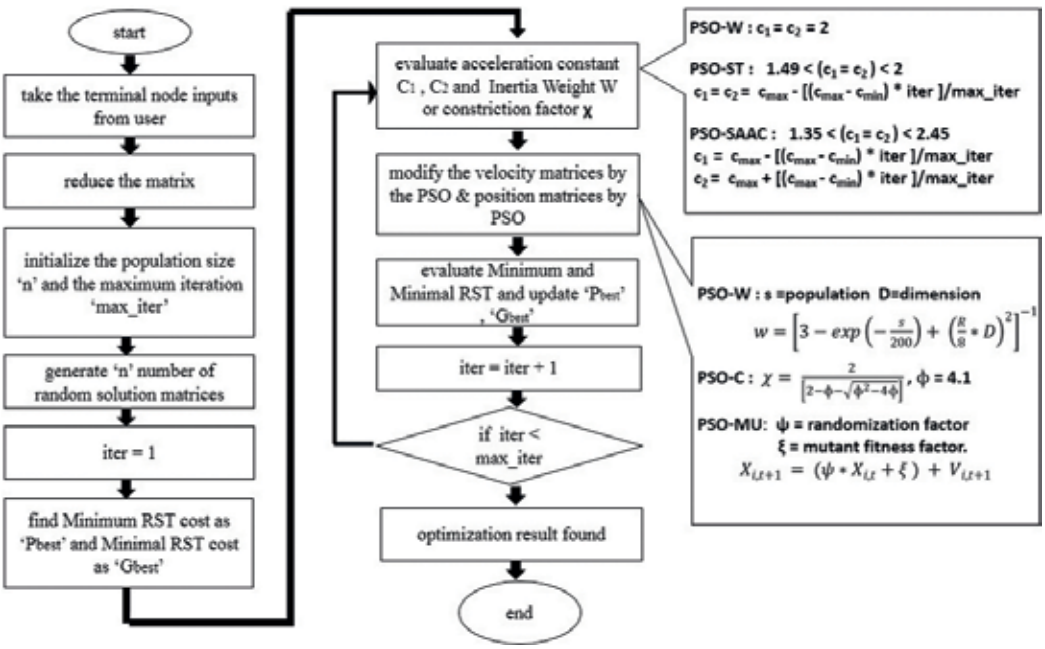

Figure 3. Flow chart of PSO algorithm and its variants.

http://dx.doi.org/10.5772/intechopen.72811

is minimum among all $MST(G_i)$ is calculated. PSO parameter are initialized and maximum iterations are fixed. Evaluation of p_{best} and g_{best} values are made using PSO velocity equations with acceleration coefficient tuned either in linear decreasing or in self-adaptive mode as described earlier and the corresponding position equation either in classical mode or with mutation factor is evaluated. When maximum iteration is reached the corresponding g_{best} value generated or the best swarm particle, is the optimized RMST cost. The optimized RMST cost on termination of the PSO algorithm is the minimum overall length of the interconnected terminal nodes in the VLSI system and thereby minimum wire length routing path of VLSI layout is achieved. The flow chart of the PSO algorithms and its variants are shown in **Figure 3**.

The pseudo code of the PSO algorithm and modifications for implementation is given below.

Input:

Search space and terminal nodes are defined

Swarm size and Max-iterations are defined

Initialization:

Generate an initial population of particles $X_i = \{X_1, X_3, \ldots\ldots, X_n\}$

Calculation of $f(X|\square|i)$ *and* $\text{MIN}(f(X|\square|i))$

Output:

Optimized result of $\text{MIN}(f(X|\square|i))$

Begin:

While (t < max iter)

Evaluate c_1 *and* c_2 *according to any of the variants mentioned in this chapter.*

Evaluate Inertia Weight (w) as in Eq. (4) *or evaluate Constriction Factor* (χ) *as in* Eq. (6)

Set $p_{best} = f(X|\square|i)$ *and* $g_{best} = \text{MIN}(f(X|\square|i))$

for i=1: n (for particles)

Calculate particle velocity $V_{i,t+1}$ *according to the velocity equation as in* Eq. (3) *or* Eq. (5)

Update the particle position $X_{i,t+1}$ *in accordance to position equation as in* Eq. (2) *or*

Update the particle position as in Eq. (7)

Evaluate $f(X|\square|i)$ *and* $\text{MIN}(f(X|\square|i))$

Update p_{best} *and* g_{best}

end for n

$t = t + 1$

end while

Post processing the results and visualization

5. Experimental results and discussion

Two coordinate sets of 15 terminal nodes are randomly created based on varied distribution topology of terminal nodes in VLSI system on a defined two dimensional 100 × 100 search space. Coordinate sets for nearly Uniform distribution and Bivariate distribution are graphically represented in **Figures 4** and **5** respectively.

Experiments on all the algorithms are performed 25 times for each of these coordinate sets of varied distribution topologies in VLSI system. The population size of the swarms has been set as 100 and maximum iteration of 75 is used for all the algorithms.

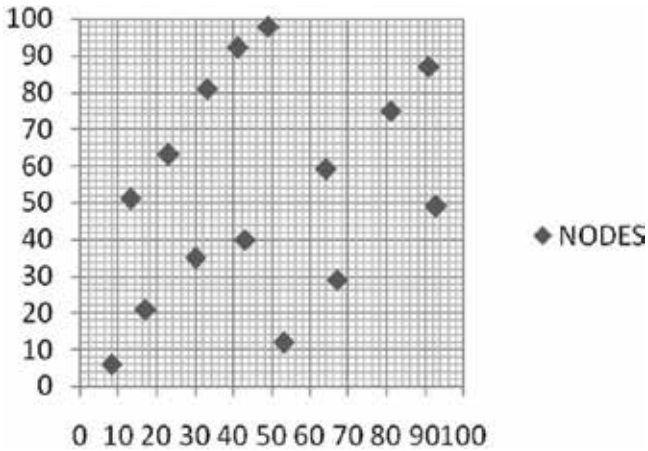

Figure 4. Nearly uniform distribution of terminal nodes on 100 × 100 search space.

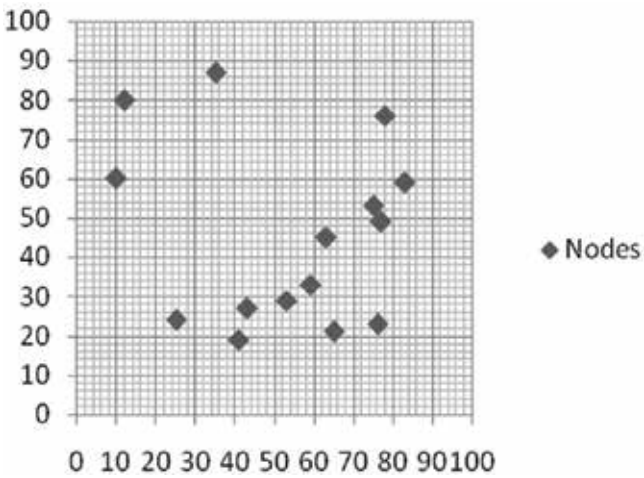

Figure 5. Bivariate distribution of terminal nodes on 100 × 100 search space.

http://dx.doi.org/10.5772/intechopen.72811

5.1. Experiment A

Experiments on PSO-W as well as on the modified algorithms PSO-ST [20] and PSO-SAAC are performed separately for two said coordinate sets to interconnect the terminal points for each set and to return the minimum cost of interconnection correspondingly. The minimum interconnection VLSI global routing cost, the average interconnection VLSI global routing cost over the 25 simulations of the algorithms and the corresponding standard deviations are recorded. The results of average g_{best} and minimum g_{best} are summarized in **Table 1**.

For nearly uniform distribution of terminal nodes in VLSI layout, PSO-SAAC works best in compared to the other two algorithms. In SET 1 '338' is achieved as the minimum interconnection global cost value for PSO-SAAC, given in **Figure 6**. From **Table 1** it can be analyzed that for bivariate distribution of terminal nodes in VLSI layout, self-tuned acceleration constant controlling mechanism for PSO-ST outruns the other two algorithms. In random uniform distribution environment, PSO-ST generates lowest minimum interconnection cost as 253,

Test case	g_{best} value	PSO-W	PSO-ST	PSO-SAAC
SET 1	Average	354.5	348.4	341.7
	Minimum	350	343	338
SET 2	Average	256.7	254.9	257.3
	Minimum	253	253	255

Table 1. Comparison of PSO-W with PSO-ST and PSO-SAAC.

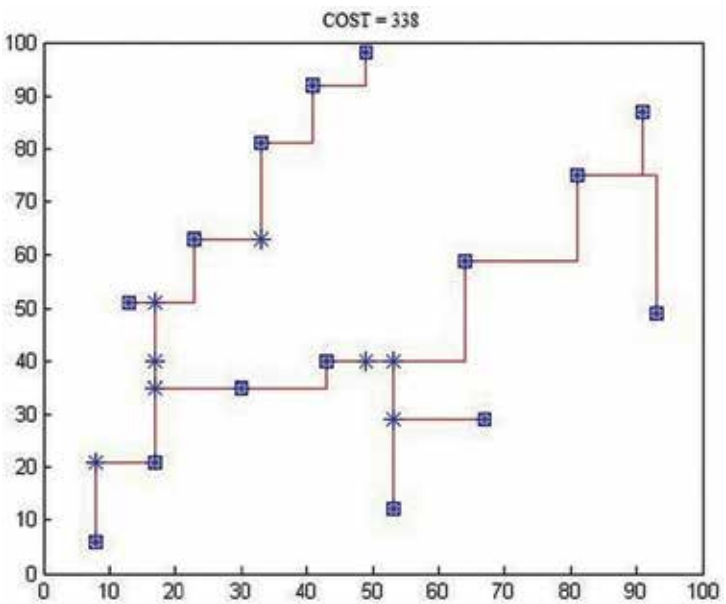

Figure 6. Minimum 'cost' Steiner tree obtained for PSO-SACC in SET 1.

given in **Figure 7**. It is also seen that acceleration constant tuning mechanism of PSO improves the average interconnection cost of the VLSI global best parameter.

So, it can be safely stated that for nearly uniform distribution PSO-SAAC and for increased random bivariate distributions PSO-ST reduces the cost of RMST, constructed by interconnecting the terminal nodes. So RSMT problem of graphs can be effectively managed and thereby the VLSI interconnect length is reduced to a great extent.

It is also observed that from **Table 2**, that standard deviation value for PSO-SAAC is lowest for SET 1 whereas PSO-ST achieves lowest standard deviation value for SET 2. This implies that for nearly uniform and less random distribution, self-adaptive mechanism of PSO ensures more consistency while self-tuned mechanism of PSO is more consistent in case of highly random distribution of terminal nodes in the defined search space.

5.2. Experiment B

The experiments are performed first on weighted PSO (PSO-W) and then on PSO with constriction factor (PSO-C) and lastly on PSO with mutation algorithm (PSO-MU) for all two

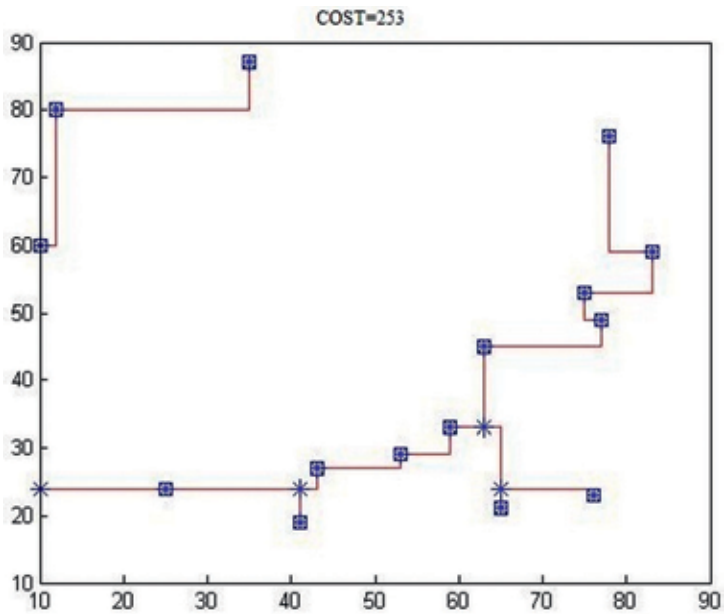

Figure 7. Minimum 'cost' Steiner tree obtained for PSO-ST in SET 2.

Test case	PSO-W	PSO-ST	PSO-SAAC
SET 1	7.77	0.71	5.41
SET 2	1.94	1.88	4.12

Table 2. Standard deviation of g_{best} values.

http://dx.doi.org/10.5772/intechopen.72811

coordinate set which have been considered. The results of minimum interconnect cost, average cost and average execution time of all algorithms are recorded on **Table 3**.

The Minimum Rectilinear Spanning Tree (RMST) for minimum interconnect cost generated for the said VLSI topologies in case of all algorithms are shown in **Figures 8**, **9**, and **10**. It reveals that the algorithm PSO-MU generate lowest minimum global best value as well as minimum mean value in all two coordinate sets. This indicates that this algorithm PSO-MU, in comparison to PSO-W and PSO-C, ensure efficient VLSI global routing cost minimization and better convergence.

It is observed from **Table 3** that for Coordinate Set 1, g_{best} value of PSO –MU is found to be 329 where execution time of this algorithm is much greater than PSO-W. Runtime of PSO-C is found to be 101.51 compared to 85.48 for PSO-MU algorithm. This implies that PSO-MU

Test case	g_{best} value	PSO-W	PSO-C	PSO-MU
SET 1	Average	354.5	350.4	336.8
	Minimum	350	345	329
	System time	52.825	101.51	85.48
SET 2	Average	256.7	256	250.4
	Minimum	253	254	248
	System time	49.05	86.01	66.96

Table 3. Comparison of PSO variants over average, minimum g_{best} value and system time.

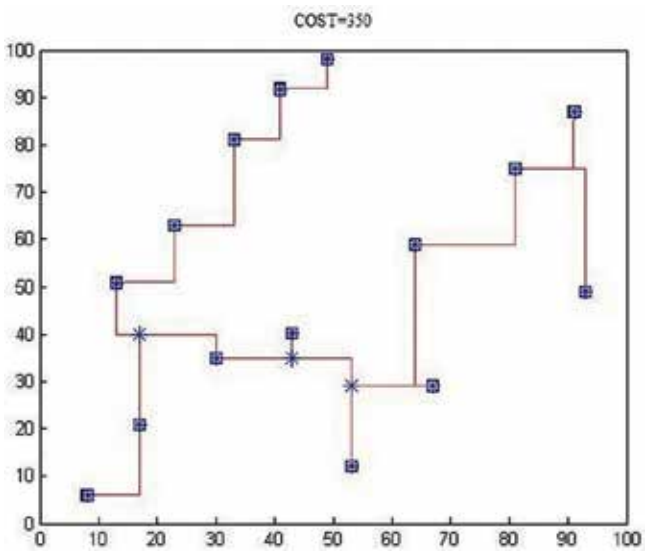

Figure 8. Minimum 'interconnection cost' Steiner Tree obtained for PSO-W in SET 1.

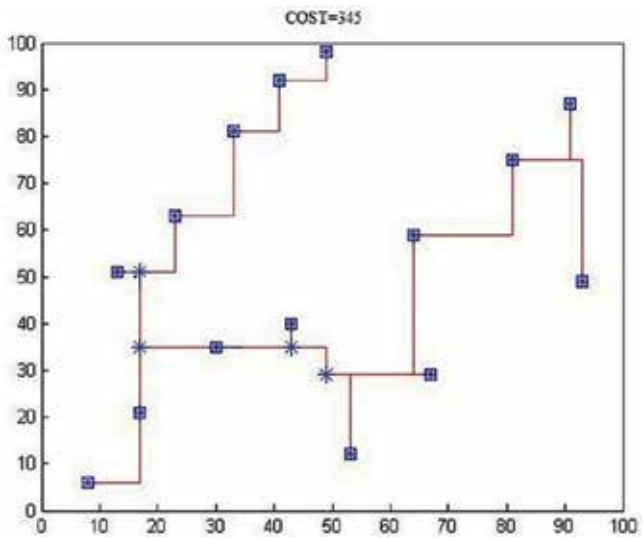

Figure 9. Minimum 'interconnection cost' Steiner Tree obtained for PSO-C in SET 1.

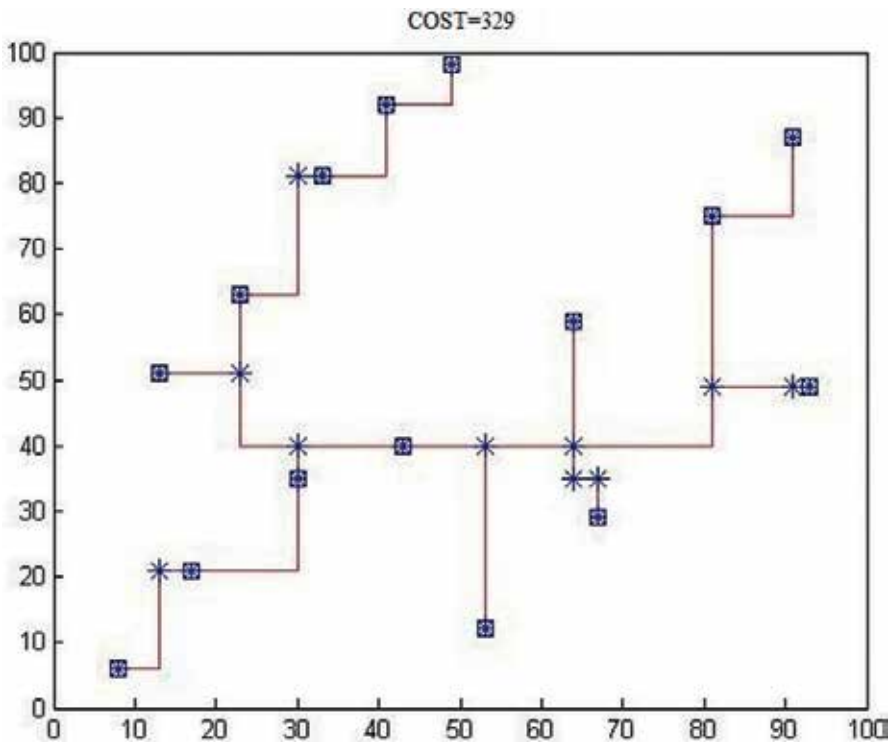

Figure 10. Minimum 'interconnection cost' Steiner Tree obtained for PSO-MU in SET 1.

Test case	PSO-W	PSO-C	PSO-MU
SET 1	7.77	0.71	5.65
SET 2	1.94	1.88	3.83

Table 4. Standard deviation of g_{best} values.

algorithm outperforms the performance of conventional PSO-W and PSO-C algorithm while reducing the Timing budget with respect to PSO-C algorithm in context to VLSI global routing.

In order to analyze the consistency of these algorithms, the standard deviation (SD) values are calculated for all algorithms on each of the two coordinate sets and are recorded in the **Table 4**. SD value of PSO-C is found to be 0.71 and 2.25 for the two coordinate sets. These values are much lower compared to all other SD values of PSO-W and PSO-MU ensuring robustness although sacrificing system execution time of the algorithm independent of the distribution complexities of the search space in VLSI layout. This implies that PSO-C although generates higher value of global routing interconnection cost as well as system execution time compared to PSO-W and PSO-MU, it exhibits robustness of the algorithm throughout all varied distribution topologies of the terminal nodes in the said VLSI layout.

6. Conclusion

This chapter intends variants developed on Particle Swarm Optimization algorithm to resolve the global routing problem in VLSI domain. Simultaneously the controlling of acceleration constant in PSO has been verified for the VLSI routing problem. Lastly, a proportional analysis is done amongst the pre mentioned algorithms beside three variants of PSO, which have been recognized as decent routing algorithms in VLSI design. Researches are piloted to inspect the optimization property, rate of convergence, computational time and robustness of the algorithms including the ways by which algorithms work proficiently in problem space with dissimilar distributive topologies of VLSI layout.

The outcomes demonstrates that from the standpoint of topologically dissimilar problem spaces of VLSI domain, the general performance of PSO-ST [20] is very agreeable, however PSO-SAAC executes finest in an approximately uniform distributed problem space. It has also been observed that the performance of PSO-C and PSO-MU are unhampered of the diverse distribution of VLSI global routing problem space. The performance of the algorithm PSO-MU preserves a balance between the optimization and convergence rate. Although PSO-MU is realized to be steady in random problem space [22], PSO-C is appeared to be the best algorithm in the perspective of robustness.

Therefore the chapter indicated the exclusive merits and demerits of the PSO algorithm and its variants, well-matched for solving the wire-length minimization problem of global routing in VLSI physical design. It is projected that in the situation of VLSI global routing optimization, the paradigm of hybridization with essence of genetics can contest with the functioning of PSO conventional ones and can exhibit enhanced performance. Hence the global routing problem

in VLSI can be competently managed by contemporary PSO meta-heuristics and by hybridization of distinct swarm intelligence.

Author details

Subhrapratim Nath[1]*, Jamuna Kanta Sing[1] and Subir Kumar Sarkar[2]

*Address all correspondence to: suvro.n@gmail.com

1 Department of CSE, Jadavpur University, India

2 Department of ETCE, Jadavpur University, India

References

[1] Moore GE. Cramming more components onto integrated circuits. Proceedings of the IEEE. 1998;**86**:82-85. DOI: S 0018-9219(98)00753-1

[2] Bollinger H. A mature DA system for PC layout. In: Proceedings of First International Printed Circuit Conference. Lo Alamitos: IEEE Computer Society Press; 1979. pp. 85-99

[3] Dees WA, Karger PG. Automated rip-up and reroute techniques. In: Proceedings of Design Automation Conference. Piscataway: IEEE Press; 1982. pp. 432-439. DOI: 10.1109/DAC.1982.1585535

[4] Ho J-M, Vijayan G, Wong CK. New algorithms for the rectilinear Steiner tree problem. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 1990;**9**(2):185-193. DOI: 10.1109/43.46785

[5] Beni G, Wang J. Swarm intelligence in cellular robotic systems. In: Dario P, Sandini G, Aebischer P, editors. Robots and Biological Systems: Towards a New Bionics? NATO ASI Series (Series F: Computer and Systems Sciences). Berlin/Heidelberg: Springer; 1993. pp. 703-712. DOI: 10.1007/978-3-642-58069-7_38

[6] Eberhart R, Kennedy J. A new optimizer using particle swarm theory. In: Sixth International Symposium on Micro Machine and Human Science, 4-6 October 1995, IEEE: Nagoya; 2002. pp. 39-43. 10.1109/MHS.1995.494215

[7] Dong C, Wang G, Chen Z, Sun S, Wang D. A VLSI routing algorithm based on improved DPSO. In: IEEE International Conference on Intelligent Computing and Intelligent Systems ICIS. Vol. 1. IEEE; 2009. pp. 802-805. DOI: 10.1109/MHS.1995.494215

[8] Ayob MN, Yusof ZM, Adam A, Abidin AFZ, Ibrahim I, Ibrahim Z, Hani MK. A particle swarm optimization approach for routing in VLSI. In: Second International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN). IEEE; 2010. pp. 49-53. DOI: 10.1109/CICSyN.2010.42

[9] Liu G, Chen G, Guo W, Chen Z. DPSO-based rectilinear Steiner minimal tree construction considering bend reduction. In: Proceedings of Seventh International Conference on Natural Computation (ICNC). Vol. 2. IEEE; 2011. pp. 1161-1165. DOI: 10.1109/ICNC.2011.6022221

[10] Shen Y, Liu Q, Guo W. Obstacle-avoiding rectilinear Steiner minimum tree construction based on discrete particle swarm optimization. In: Proceedings of Seventh International Conference on Natural Computation (ICNC). Vol. 4. IEEE; 2011. pp. 2179-2183. DOI: 10.1109/ICNC.2011.6022440

[11] Arora T, Moses ME Ant colony optimization for power efficient routing in Manhattan and non-Manhattan VLSI architectures. In: Proceedings of Swarm Intelligence Symposium, SIS '09, March-April 2 2009. IEEE, pp.137-144. DOI: 10.1109/SIS.2009.4937856

[12] Ayob MN, Hassan F, Ismail AH, Basri HH, Azmi MS, Abidin AFZ A firefly algorithm approach for routing in VLSI. In: Proceedings of IEEE Symposium on Computer Applications and Industrial Electronics (ISCAIE), 3-4 December 2012 pp.43-47. DOI: 10.1109/ISCAIE.2012.6482066

[13] Shi Y, Eberhart RC. Empirical study of particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation; 6-9 July 1999; IEEE: Washington, DC; 2002. p. 1945-1950. DOI: 10.1109/CEC.1999.785511

[14] Clerc M, Kennedy J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation. 2002;**6**(1):58-73. DOI: 10.1109/4235.985692

[15] Garey MR, Johnson DS. The rectilinear Steiner tree problem is NP-complete. SIAM Journal on Applied Mathematics. 1977;**32**(4):826-834. DOI: 10.1137/0132071

[16] Carlisle A, Dozier G. An off-the-shelf PSO. In: Proceedings of the Workshop on Particle Swarm Optimization. Indianapolis: Purdue School of Engineering and Technology, IUPUI; 2001. pp. 1-6. DOI: 10.1.1.589.485

[17] Valle YD, Venayagamoorthy GK, Mohagheghi S, Hernandez J-C, Harley RG. Particle swarm optimization: basic concepts, variants and applications in power systems. IEEE Transactions on Evolutionary Computation. 2008;**12**(2):171-195. DOI: 10.1109/TEVC.2007.896686

[18] Eberhart RC, Shi Y. Particle swarm optimization: developments, applications and resources. In: Proceedings of the 2001 Congress on Evolutionary Computation; 27-30 May 2001 IEEE: Seoul; 2002. pp. 81-86. DOI: 10.1109/CEC.2001.934374

[19] Jordehi AR, Jasni J. Parameter selection in particle swarm optimisation: A survey. Journal of Experimental and Theoretical Artificial Intelligence. 2013;**25**(4):527-542. DOI: 10.1080/0952813X.2013.782348

[20] Ghosh S, Nath S, Sarkar SK. PSO algorithm with self tuned parameter for efficient routing in VLSI design. In: International Conference on Futuristic Trends in Computing and Communication; 29th April 2015; Chennai; 2015. SSRG International Journal of Electronics and Communication Engineering. pp. 60-63. DOI: ISSN: 2348–8549

[21] Hassan R, Cohanim B, Weck OD, Venter G. A comparison of particle swarm optimization and the genetic algorithm. In: 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference; 18-21 April 2015. American Institute of Aeronautics and Astronautics: Austin; 2015. pp. 1-13. DOI: 10.2514/6.2005-1897

[22] Nath S, Ghosh S, Sarkar SKA. Novel approach to discrete particle swarm optimization for efficient routing in VLSI design. In: 4th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions); 2-4 September 2015. IEEE: Noida; 2015. DOI: 10.1109/ICRITO.2015.7359375

Chapter 6

Solution of Combined Economic Emission Dispatch Problem with Valve-Point Effect Using Hybrid NSGA II-MOPSO

Arunachalam Sundaram

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.72807

Abstract

This chapter formulates a multi-objective optimization problem to simultaneously minimize the objectives of fuel cost and emissions from the power plants to meet the power demand subject to linear and nonlinear system constraints. These conflicting objectives are formulated as a combined economic emission dispatch (CEED) problem. Various meta-heuristic optimization algorithms have been developed and successfully implemented to solve this complex, highly nonlinear, non-convex problem. To overcome the shortcomings of the evolutionary multi-objective algorithms like slow convergence to Pareto-optimal front, premature convergence, local trapping, it is very natural to think of integrating various algorithms to overcome the shortcomings. This chapter proposes a hybrid evolutionary multi-objective optimization framework using Non-Dominated Sorting Genetic Algorithm II and Multi-Objective Particle Swarm Optimization to solve the CEED problem. The hybrid method along with the proposed constraint handling mechanism is able to balance the exploration and exploitation tasks. This hybrid method is tested on IEEE 30 bus system with quadratic cost function considering transmission loss and valve point effect. The Pareto front obtained using hybrid approach demonstrates that the approach converges to the true Pareto front, finds the diverse set of solutions along the Pareto front and confirms its potential to solve the CEED problem.

Keywords: multi-objective optimization, economic emission dispatch, Pareto optimality, NSGAII, MOPSO, B-loss coefficients

1. Introduction

In order to operate the power system economically and also to protect the environment from pollution the power system operator has to carry out optimal scheduling of active power to simultaneously minimize the fuel cost and the emissions from the fossil fuel-fired power plants.

These objectives are desirable to obtain great economic benefit [1] and to reduce the nitrogen oxide (NO_x), sulfur oxide (SO_x) and carbon dioxide (CO_2) pollutants which cause harmful effect on human beings [2]. These conflicting objectives can be formulated as a multi-objective combined economic emission dispatch (CEED) problem. This CEED problem can be solved using traditional mathematical programming techniques such as lambda iteration, gradient search [1] and can also be solved using modern heuristics optimization techniques. The numerous advantages of solving the CEED problem using heuristic optimization methods compared to the traditional mathematical programming techniques are they are population-based, do not require any derivative information, do not use gradient information in search process, use stochastic operators in search process, they are simple to implement and flexible, have inbuilt parallel architecture and they are scalable and are also computationally quick.

A single optimal solution cannot be obtained for a multi-objective CEED problem which simultaneously minimizes the conflicting objectives of fuel cost and emission. Thus the simultaneous minimization of conflicting objectives in a multi-objective optimization problem (MOP) gives rise to a set of tradeoff solution called as Pareto-optimal (PO) solutions [3] which needs further processing to arrive at a single preferred solution. In literature domination based framework using multi-objective evolutionary algorithms (MOEA) which simultaneously minimizes the fuel cost and emission have been employed to solve the CEED problem. These population-based approaches can obtain the multiple non dominated solutions in a single simulation run. These non-dominated solutions portray the tradeoff between fuel cost and emission objectives of CEED problem. Modern meta-heuristic optimization algorithms like Genetic Algorithm [4, 5], Biogeography Based Optimization [6], Particle Swarm Optimization [7], Bacterial Foraging Algorithm [8], Scatter Search [9], Teaching Learning Based Optimization [10], Differential Evolution [11] and Harmony Search Algorithm [12] have been developed and successfully implemented to solve this complex, highly nonlinear, non-convex CEED problem.

The multiple objective CEED problem can also be transformed into a single objective problem using a weighted sum approach and h parameter values. The h parameters are used to overcome the dimensionality problem when combining multi-objectives and the converted single objective problem is then solved using evolutionary algorithms [13–15]. Another technique to solve CEED problem without the h parameter is to normalize the fuel cost and emission components [6] and solve the single objective function using evolutionary algorithms (EA). In these approaches for the chosen value of weights will give one particular PO solution at a time. However, the disadvantage of these methods is that it requires multiple runs to find the set of PO solutions.

Each evolutionary algorithm has its own characteristics and merits; therefore it is natural to think of integrating these different algorithms to handle a complex problem like CEED. In the research field of Evolutionary Algorithms merging of two or more optimization algorithms into a single framework is called hybridization. In [16–21] hybrid multi-objective optimization algorithms have been successfully applied to solve CEED, various complex engineering problems, and standard test functions. The results indicate that the hybrid algorithms are effective, can exchange elite knowledge within the hybrid framework, can do parallel processing, can improve the exploration and exploitation capabilities and can yield more favorable performance than any single algorithm.

http://dx.doi.org/10.5772/intechopen.72807

In order to obtain a globally optimal solution without being trapped in local optima requires a tradeoff between exploration and exploitation task in the search process. Exploration phase in any algorithm is important to search every part of the solution domain to provide an estimate of the global optimal solution. On the other hand exploitation phase in any algorithm is important to improve the best solutions found so far by searching in their neighborhood. In this chapter, a hybrid framework using Non-Dominated Sorting Genetic Algorithm II (NSGA II) [22] and Multi-objective Particle Swarm Optimization (MOPSO) [23] is used to solve the CEED problem. This hybrid framework integrates the desirable features of the NSGA II and MOPSO while curbing their individual flaws. These population-based approaches use different techniques for exploring the search space and when they are combined will improve the tradeoff between the exploration and exploitation tasks to converge around the best possible solutions. The main purpose of this hybridization technique is to obtain a well-spread and well-diverse PO solution. When the proposed hybrid algorithm is used to solve the highly complex CEED problem the PO solution is obtained in less number of iteration and is also computationally fast when compared to MOPSO.

The rest of the chapter is organized as follows. The next section formulates the CEED problem. In Section 3, the transmission loss handling procedure and the constraint handling procedure is explained. In Section 4 the short review of NSGA II and MOPSO is provided. Section 5 is devoted to explaining the hybrid algorithm. In Section 6 the hybrid algorithm is applied on standard IEEE 30 bus systems and it also discusses the simulation results. Finally, the conclusion is drawn in Section 7.

2. Formulation of combined economic emission dispatch (CEED) problem

The combined economic emission dispatch problem has two conflicting objectives. The first objective can be stated as determining the optimal power generation schedule from a set of online generating units to satisfy the load demand subject to several physical and operational constraints to minimize the fuel cost. The second objective can be stated as determining the optimal power generation schedule from a set of online generating units to satisfy the load demand to minimize the pollutant emissions produced by the generating units. Both the conflicting objectives have to be minimized at the same time because operating the system with minimum cost will result in higher emission and considering only the minimum environmental impact is not practical which results in high production cost of the system. This section formulates the objective functions of the CEED problem along with equality and inequality constraints to maintain rigorous standards to meet the practical requirements of the power system. The goal of this chapter is to find the Pareto-optimal solutions of the CEED problem which minimize both these objectives subject to constraints. The mathematical formulation is as follows.

2.1. Objective functions of CEED problem

The general formulation for a multi-objective optimization problem (MOOP) is to minimize the number of objective functions simultaneously. A general mathematical model is represented as follows [21]:

$$\text{Minimize } f(x) = \{f_1(x), f_2(x), \cdots f_m(x)\}, x \in \mathcal{D} \tag{1}$$

where $f(x)$ represents the vector of objectives and $f_i(x), i = 1, 2, \cdots, m$ is a scalar decision variable which maps decision variable x into objective space $f_i = \Re^n \to \Re$. The n-dimensional variable x is restricted to lie in a feasible region $\mathcal{D}$ which is constrained by j in-equality constraint and k equality constraint, i.e.

$$\mathcal{D} = \left\{x : g_j(x) \leq 0, h_k(x) = 0, j = 1, 2, \cdots J; k = 1, 2, \cdots, K\right\} \tag{2}$$

The decision variable x can be written more suitably as

$$x = [x_1, x_2, x_3, \cdots, x_n]^T \tag{3}$$

where T is the transposition of the column vector to the row vector. The decision variables are restricted to take a value within a lower $x_i^{(\min)}$ and upper $x_i^{(\max)}$ bounds. These bounds are called the decision space [3].

In MO CEED problem the number of objectives $m = 2$.The mathematical model of CEED is represented as follows:

$$\text{Minimize } f(x) = \{f_1(x), f_2(x)\}, x \in \mathcal{D} \tag{4}$$

subject to power balance equality constraints $h(x)$ and bounds. The function $f_1(x)$ represents the minimization of total fuel cost function and the function $f_2(x)$ represents the minimization of the emissions from the fossil fuel fired plants. The decision variable x consists of the real power generation of the n generating units and can be written as

$$x = \left[Pg_1, Pg_2, Pg_3, \cdots, Pg_n\right]^T \tag{5}$$

where Pg_i is the real power output of the i^{th} generator.

Power plants commonly have multiple valves that are used to control the power output of the units. In a practical generating unit, when steam admission valves in thermal units are first opened, a sudden increase in losses is registered which results in ripples in the cost function. In order to model these ripples accurately, sinusoidal functions are added to the quadratic cost function [24]. The resulting cost function contains higher order nonlinearity and makes the problem non-differentiable and non-convex. Hence there are two versions of the fuel cost function, the quadratic function represented by $f_1(x)$ and the combination of quadratic and a sinusoidal (valve-point) function represented by $f_{1,V}(x)$. The two versions of the fuel cost functions are given below

$$f_1(Pg) = \sum_{i=1}^{n} a_i Pg_i^2 + b_i Pg_i + c_i \tag{6}$$

$$f_{1,V}(Pg) = \sum_{i=1}^{n} a_i Pg_i^2 + b_i Pg_i + c_i + \left|e_i \sin\left\{f_i\left(Pg_i^{min} - Pg_i\right)\right\}\right| \tag{7}$$

where a_i, b_i, c_i represent the cost coefficients of the generator *i*. e_i and f_i are coefficients to model the effect of valve point of the generator *i*.

The second objective function $f_2(x)$ is an emission function which takes into account the major pollutants caused by the fossil fuel fired power plants. The main pollutants from the power plants are the sulfur oxides and nitrogen oxides. The sulfur oxide emissions are proportional to the fuel consumed by the power plants and have the same form as that of the fuel cost function given by (6). The sulfur oxide emission function can be stated as follows [7].

$$f_{2,so}(Pg) = \sum_{i=1}^{n} S_{i,1} + S_{i,2}Pg_i + S_{i,3}Pg_i^2 \tag{8}$$

The nitrogen oxides emissions are difficult to evaluate as the nitrogen is available in air and also in the fuel. The production of nitrogen gas is related to boiler temperature and air content. The modeling of the nitrogen oxides consists of straight lines and exponential terms. The nitrogen oxides emission function can be stated as follows

$$f_{2,No}(Pg) = \sum_{i=1}^{n} N_{i,1} + N_{i,2}Pg_i + N_{i,3}e^{N_{i,4}Pg_i} \tag{9}$$

The total emission function is obtained by adding the coefficients of (8) and (9) which gives the combination of the mixture of sulfur oxides and nitrogen oxides pollutants [7]. The total emission function can be stated as follows

$$f_2(Pg) = \sum_{i=1}^{n} \left[10^{-2}\left(\alpha_i + \beta_i Pg_i + \gamma_i Pg_i^2\right) + \eta_i e^{\delta_i Pg_i}\right] \tag{10}$$

The total emission function given by (10) has a quadratic term and an exponential term which makes the function highly nonlinear. In (10) α_i, β_i, γ_i, η_i, δ_i are the emission coefficients of the generator *i*. The modeling of the emission function is very important because according to the Amendments of the Clean Air Act regulatory agencies might decide to limit power plant emission in the areas where there are high concentrations of harmful contaminants.

2.2. Active power balance equality constraint and bounds

In order ensure that the total real power generation exactly match with the total load demand *Pd* and transmission loss *Pl* in the system a power balance equality constraint given in (11) should be satisfied.

$$h(x) = \sum_{i=1}^{n} Pg_i - Pd - Pl = 0 \tag{11}$$

The transmission losses in the power network are function of *Pg* and can be represented using B-matrix coefficients (Kron's loss formula [1]) as follows

$$Pl(Pg) = \sum_{i=1}^{n}\sum_{j=1}^{n} Pg_i B_{ij} Pg_j + \sum_{i=1}^{n} B_{0i}Pg_i + B_{00} \tag{12}$$

where B_{ij}, B_{0i}, B_{00} are transmission loss coefficients. There are instances in literature where the power losses in the system is neglected and the power balance equation given by (11) is curtailed as follows

$$h(x) = \sum_{i=1}^{n} Pg_i - Pd = 0 \tag{13}$$

The above equations given by (11) and (13) are most common form of power balance equation found in the literature.

The power output of each generator i should lie within its minimum limit (Pg_i^{min}) and maximum limit (Pg_i^{max}) given by

$$Pg_i^{min} \leq Pg_i \leq Pg_i^{max}; i = 1, 2, 3 \cdots, n \tag{14}$$

2.3. Combined economic emission dispatch

The purpose of the CEED problem is to determine the Pareto-optimal real power generation vector $x^* = \left[Pg_1^*, Pg_2^*, Pg_3^*, \cdots, Pg_n^*\right]^T$ that minimize the two conflicting objective given by (7) and (10) while satisfying the real power equality constraint given by (11) and the bounds given by (14). The bi-objective CEED problem can be formulated as

$$\text{Minimize } f = \left[f_{1,V}(Pg), f_2(Pg)\right] \tag{15}$$

In MO CEED problem, the economic and emission objectives will conflict with each other and is not possible to satisfy them simultaneously. There is no way of improving these objectives without degrading at least one of these objectives and the resulting set of non-dominated solutions thus obtained are called Pareto-optimal set. The objective function values of all elements in the PO set in the objective space constitute the Pareto front. When the sufficient number of PO solutions is available for the CEED problem then it is possible to find a convex curve containing these solutions to produce the Pareto front. The two main goals of MO CEED problem:

1. Find a set of non-dominated solutions which lie on the Pareto-optimal front
2. Find a wide spread of non-dominated solutions to represent the entire range of the Pareto-optimal front.

3. Constraint handling mechanism

At any stage of the algorithm whenever a new population is being generated it is very important to make sure that the population lies within the decision space. While solving the CEED problem this implies that the population should satisfy the equality constraints and bounds. If the transmission losses are neglected than the k^{th} variable of the candidate solution Pg_k can be calculated by subtracting the sum of the power generations (excluding the

k^{th} variable) $\sum_{i=1}^{n-1} Pg_i$from the power demand Pd. If the power transmission losses are considered, to determine Pg_k and to maintain the equality constraint becomes hard. It is done using the following steps.

Step 1. Update the variables belonging to the set α_n by normal optimization process of an evolutionary algorithm.

$$Pg_i = Pg_i^{min} - \text{rand}*\left(Pg_i^{min} - Pg_i^{max}\right); i \in \alpha_n \tag{16}$$

Here *rand* is a uniformly distributed random number in the range *of* $[0, 1]$. The set α_n contains all the integers in the range $[1, n]$ except k, where k is a randomly generated integer which lies in the range of $[1, n]$

Step 2. If updating of the variables is carried out using any other technique then regulate the updated variables which violate the lower bounds as $Pg_i = Pg_i^{min}; i \in \alpha_n$. Regulate the updated variables which violate the upper bounds as $Pg_i = Pg_i^{max}; i \in \alpha_n$.

Step 3. Obtain the value of the k^{th} variable of the candidate solution Pg_k by solving the following quadratic equation (17) whose coefficients are associated with the variables belonging to the set α_n and the transmission loss coefficients [7]. To improve the potential candidate solution and also to improve the flexibility and diversity of the optimization algorithm the value of k is randomly generated integer between 1 and n.

$$\begin{gathered} B_{kk}Pg_k^2 + \left(2\sum_{i\in\alpha_n} B_{ki}Pg_i + B_{0k} - 1\right)Pg_k + \\ \left(Pd + \sum_{i\in\alpha_n}\sum_{j\in\alpha_n} Pg_i B_{ij} Pg_j + \sum_{i\in\alpha_n} B_{0i}Pg_i - \sum_{i\in\alpha_n} Pg_i + B_{00}\right) = 0 \end{gathered} \tag{17}$$

Out of the two roots of the quadratic equation (17), one root will be selected as the value of the variable Pg_k using the following procedure. If both the roots of the quadratic equation lie within the bounds then the root which has the minimum value is selected. If only one root lies within the bounds, this root is selected as the value of Pg_k and the other root which lies outside the bounds is neglected. If both the roots lay outside the bounds the value of Pg_k is set equal to Pg_k^{min}.

Step 4. Calculate the residue P_{RD} by subtracting the total system demand Pd and the total system transmission loss Pl from the sum of the total power generation $\sum_{i=1}^{n} Pg_i$. If $|P_{RD}| < tol$, then go to step 7; otherwise go to step 5. Here, tol is the demand tolerance usually set as $0.001\ p.u.$

Step 5. Recalculate Pg_i using Eq. (16).

Step 6. Repeat step 3, step 4 and step 5 until $|P_{RD}| < tol$. This step will ensure that the candidate solution will always lie within the decision space.

Step 7. Stop the constraint handling procedure.

The main purpose of this constraint handling mechanism is to increase the flexibility and diversity of the algorithm and to make sure that the candidate solution generated at any point of the algorithm always lies within the decision space.

4. NSGA II and MOPSO algorithms for solving CEED problem

Several Evolutionary Multi-objective (EMO) algorithms like NSGAII, MOPSO, SPEA 2 (Strength Pareto Evolutionary Algorithm), GDE 3 (Generalized Differential Equation) have been designed and used in solving numerous complex real word problems involving two or more objectives. All these algorithms can find the multiple Pareto-optimal solutions in a single run. Out of all these available algorithms, two of the widely used reliable methods for solving bi-objective optimization problems are the NSGA II and MOPSO. This section provides the review of these two EMO algorithms.

NSGA II was proposed in [22] as an improvement of the NSGA proposed in [25]. This NSGA II algorithm was the revised version of NSGA to overcome the following criticisms:

- Computational complexity associated with non-dominated sorting.
- Lack of elite-preserving strategy.
- Lack of maintaining diversity among obtained solutions.

The NSGA II algorithm is very efficient for solving multi-objective optimization problems since it incorporates an efficient elitism preserving technique using non-domination sorting. The population is ranked based on non-domination sorting before the selection is performed. All non-dominated individuals are classified into one category. Another layer of non-dominated individuals are considered after the group of classified individuals are ignored. This process is continued until all individuals in the population are classified. NSGA II also uses a mechanism for preserving the diversity and spread of the solutions without specifying any additional parameters (NSGA uses fitness sharing). This crowding distance operator guides the selection process towards a uniformly spread out Pareto-optimal front. ***The NSGA II algorithm for solving the CEED problem is stated below:***

- **Specify the parameters for the CEED problem**
  - The total demand of the power system *Pd*
  - Fuel cost and emission coefficients for each generating unit
  - B matrix coefficients for transmission loss calculations
  - Number of decision variables *nVar*
  - Lower bounds of the decision variables *VarMin*
  - Upper bounds of the decision variables *VarMax*
- **Specify the parameters for NSGA II Algorithm**
  - Population Size nPop

- Maximum number of iteration MaxIt
- Crossover Percentage pCrossover
- Mutation Percentage pMutation
- Mutation rate mu
- Mutation step size sigma

- **Initialize Population**
  - Generate a random nPop size population
  - Once the random population is initialized the Constraint Handling Mechanism proposed in Section 3 is carried out.
- **Evaluate the objective functions**
  - Evaluate the fuel cost objective function E and emission objective function F
- **Perform Non Domination Sorting**
- **Calculate Crowding Distance and rank the population based on Non Dominated fronts**
- **For each generation do**
  - Create offspring population
    - Selection, Crossover and Mutation
  - Apply Constraint Handling Mechanism
  - Evaluate the fuel cost objective function E and emission objective function F
  - Merge the parent and offspring population
  - Perform non domination sorting
  - Calculate crowding distance and rank based on non-domination fronts
  - Select solutions
    - Each front is filled in ascending order
    - Last front-descending order of crowding distance
  - Store the non-dominated solutions in list F_1
  - Plot the non-dominated solutions in list F_1
  - Increment generation count
- **End for**

In order to handle multiple objectives Pareto dominance is incorporated into PSO algorithm and the MOPSO algorithm is proposed in [23]. The algorithm proposed in [23] uses an external repository of particles to keep a record of the non-dominated vectors found along the search

process. At each generation, for each particle in the swarm, by using Roulette wheel selection, a leader is selected from the external repository. This leader then guides other particles towards better regions of the search space by modifying the flight of the particles. A special mutation operator is applied to the particles of the swarm and also to the range of each design variable of the problem to be solved to improve the explorative behavior of the algorithm. The value of the mutation operator is decreased during the iteration. To produce well spread Pareto fronts the MOPSO algorithm in [23] uses an adaptive grid. ***The MOPSO algorithm for solving the CEED problem is stated below:***

- **Specify the parameters for the CEED problem**
  - The total demand of the power system Pd
  - Fuel cost and emission coefficients for each generating unit
  - B matrix coefficients for transmission loss calculations
  - Number of decision variables nVar
  - Lower bounds of the decision variables VarMin
  - Upper bounds of the decision variables VarMax
- **Specify the parameters for MOPSO Algorithm**
  - Maximum number of iteration MaxIt
  - Population Size nPop
  - Repository size nRep
  - Inertia Weight w and Inertia Weight damping rate wdamp
  - Personal learning coefficient c1 and Global learning coefficient c2
  - Number of grids per dimension nGrid
  - Inflation Rate alpha, leader selection pressure beta, Deletion selection pressure gamma
  - Mutation rate mu
- **Initialize Swarm Population**
  - Generate a random swarm particles
  - Once the random particles are initialized the Constraint Handling Mechanism (Section 3) is carried out.
  - Store the values of the particles as their personal best pBest
- **Determine Domination**
  - Initialize external repository rep
  - Create grid and find grid index

- **For each generation do**
- **For each particle do**
    - Select leader from external repository
    - Update particle position and velocity
    - Apply Constraint Handling Mechanism
    - Evaluate the fuel cost objective function E and emission objective function F
    - Apply Mutation and calculate new solutions
    - Apply Constraint Handling Mechanism
    - Determine Domination
    - Update pBest
- **End for**
    - Add non dominated particles to the repository
    - Determine domination of new repository members
    - Keep only the non-dominated members in the repository
    - Update grid and grid index
    - If repository is full delete members
    - Plot the members in the external repository
    - Modify inertia weight
- **End for**

5. Hybrid NSGA II and MOPSO algorithm for solving CEED problem

The mechanism of the proposed hybrid approach for solving the CEED problem is to integrate the desirable features of NSGA II (retaining the elitism feature) and MOPSO (exploitation capability) while curbing the individual flaws (NSGAII—does not have an efficient feedback mechanism, PSO overutilization of resources). The mechanism to explore the search space differs in both the algorithms. GA uses mutation and crossover operators which will enhance the exploration task of the hybrid algorithm. The particles in PSO are influenced by their own knowledge and information shared among swarm members. PSO enhances the exploitation task of the hybrid algorithm by finding better solutions from the good ones by searching the neighborhood of good solutions. In this hybrid algorithm at every generation, the Pareto dominance of the population is computed and based on these values non dominated sorting is performed [19]. In order to avoid premature convergence, the elite upper half of the population are enhanced by NSGA II algorithm while the lower

half of the population are considered as swarm particles and are optimized by MOPSO to make them converge around the best possible solutions. ***The hybrid NSGA II-MOPSO algorithm for solving the CEED problem is stated below:***

- **Specify the parameters for the CEED problem**
- **Specify the parameters for NSGA II Algorithm**
  - Population Size *nPop*
  - Maximum number of iteration *MaxIt*
  - Crossover Percentage *pCrossover*
  - Mutation Percentage *pMutation*
  - Mutation rate *mu*
  - Mutation step size *sigma*
- **Specify the parameters for MOPSO Algorithm**
  - Repository size nRep
  - Inertia Weight w and Inertia Weight damping rate wdamp
  - Personal learning coefficient c1 and Global learning coefficient c2
  - Number of grids per dimension nGrid
  - Inflation Rate alpha, leader selection pressure beta, Deletion selection pressure gamma
  - Mutation rate mu
- **Initialize Population**
  - Generate a random nPop size population
  - Once the random population is initialized the Constraint Handling Mechanism proposed in Section 3 is carried out.
- **Evaluate the objective functions**
  - Evaluate the fuel cost objective function E and emission objective function F
- **For each generation do**
  - Perform Non Domination Sorting
  - Calculate Crowding Distance and rank the population based on Non Dominated fronts
  - Truncate and divide the population into two halves.
  - Using the upper half of the population create offspring population
    - Selection, Crossover and Mutation

- Perform Constraint Handling Mechanism
- Evaluate the fuel cost objective function E and emission objective function F
- Merge the parent and offspring population
- Perform non domination sorting
- Calculate crowding distance and rank based on non-domination fronts
- Select solutions
    - Each front is filled in ascending order
    - Last front- descending order of crowding distance
- Store the non-dominated solutions in list F_1
- Plot the non-dominated solutions in list F_1
- Position and cost of the particle are initialized from the lower half of the population
- Store the values of the particles as their personal best pBest
- Determine Domination
    - Initialize external repository rep
    - Create grid and find grid index
- For each particle do
    - Select leader from external repository
    - Update particle position and velocity
    - Constraint Handling Mechanism
    - Evaluate the fuel cost objective function E and emission objective function F
    - Apply Mutation and calculate new solutions
    - Apply Constraint Handling Mechanism
    - Determine Domination pBest
- End for
- Add non dominated particles to the repository
- Determine domination of new repository members
- Keep only the non-dominated members in the repository
- Update grid and grid index
- Modify inertia weight

- Create a new set of particles half the size nPop and fill it with the non-dominated solutions in the repository followed by the pBest
- Combine the populations of NSGA II and the new set of particles of the MOPSO
- Increment generation count

- **End for**

6. Numerical tests

In order to validate the proposed hybrid algorithm, the CEED problem was solved for IEEE 30-bus system and the results are presented in this section. The fuel cost coefficients with valve-point loading, emission coefficients, and generator limits are adapted from [26] and is given in **Table 1**. The transmission loss B-matrix coefficients are obtained by running a load flow program and is in [26] is adapted here and given in **Table 2**. The total power demand in the system is 2.834 p.u. to the base of 100 MVA. Program in MATLAB was developed for the Hybrid Algorithm to perform CEED and executed on 1.60 GHz, Intel T2050 processor, 1.5 GB RAM HP Pavilion Laptop with WINDOWS 7 operating system. Various test cases are considered to compute the Pareto front of the multi-objective CEED problem. The Pareto-optimal front is obtained using the NSGA II algorithm and also using the MOPSO algorithm given in Section 4. The Pareto front obtained from the hybrid approach given in Section 5 is then compared with the Pareto front obtained using NSGAII and MOPSO algorithm.

In case 1 the fuel cost function is modeled as a quadratic function with sine term to incorporate the valve-point effect. The transmission losses are also considered in this case. The Pareto front obtained using NSGA II, MOPSO, and Hybrid NSGAII-MOPSO is shown in **Figures 1**, **2** and **3** respectively. In all these figures there is a discontinuity in the Pareto front due to modeling of the valve point loading effect of generators.

The parameter settings for NSGA II are obtained using trial and error is as follows: $M = 2$; Population Size $nPop = 100$; Maximum number of iteration $MaxIt = 100$; Crossover Percentage

Unit i	Generation Limits		Fuel Cost Coefficients with valve point loading					Emission Coefficients				
	Pg_i^{min}	Pg_i^{max}	a_i	b_i	c_i	e_i	f_i	α_i	β_i	γ_i	η_i	δ_i
1	0.05	0.5	10	200	100	15	6.283	4.091	−5.554	6.490	$2e^{-4}$	2.857
2	0.05	0.60	10	150	120	10	8.976	2.543	−6.047	5.638	$5e^{-4}$	3.333
3	0.05	1.00	20	180	40	10	14.784	4.258	−5.094	4.586	$1e^{-6}$	8.000
4	0.05	1.20	10	100	60	5	20.944	5.326	−3.550	3.380	$2e^{-3}$	2.000
5	0.05	1.00	20	180	40	5	25.133	4.258	−5.094	4.586	$1e^{-6}$	8.000
6	0.05	0.60	10	150	100	5	18.480	6.131	−5.555	5.151	$1e^{-5}$	6.667

Table 1. Fuel costs Coefficients with valve point loading, Emission Coefficients, Generator limits of IEEE 30 bus system.

B	0.02180	0.01070	−0.00036	−0.00110	0.00055	0.00330
	0.01070	0.01704	−0.00010	−0.00179	0.00026	0.00280
	−0.00040	−0.00020	0.02459	−0.01328	−0.01180	−0.00790
	−0.00110	−0.00179	−0.01328	0.02650	0.00980	0.00450
	0.00055	0.00026	−0.01180	0.00980	0.02160	−0.00010
	0.00330	0.00280	−0.00792	0.00450	−0.00012	0.02978
*B*0	1.0731e−05	0.0017704	−0.0040645	0.0038453	0.0013832	0.0055503
*B*00	0.0014					

Table 2. B−Loss Coefficients for IEEE 30 bus test system.

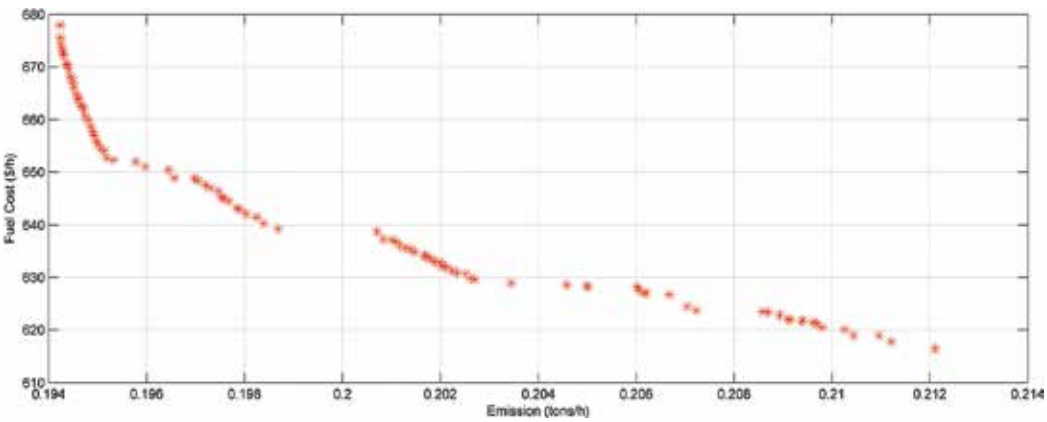

Figure 1. Pareto-optimal curve for IEEE 30 bus system obtained using NSGA II.

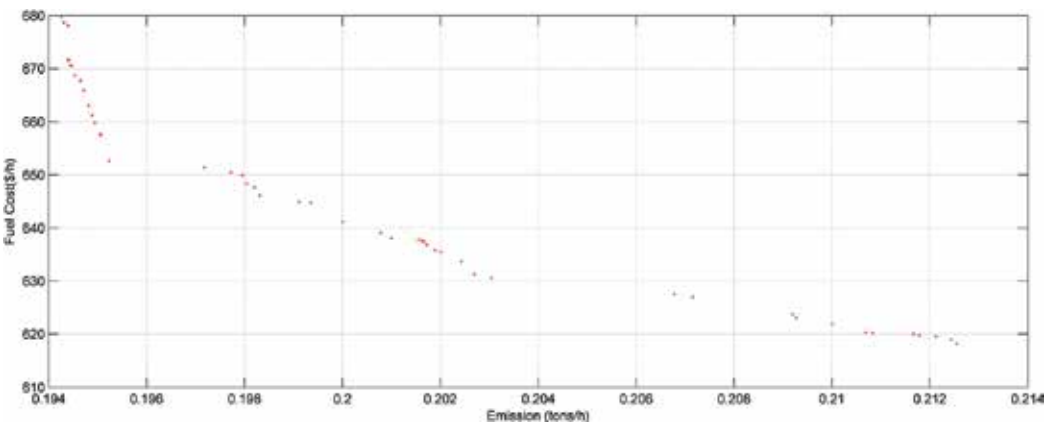

Figure 2. Pareto-optimal curve for IEEE 30 bus system obtained using MOPSO.

$pCrossover = 0.7$; Mutation Percentage $pMutation = 0.4$; Mutation rate $mu = 0.02$. The extreme points of the Pareto front and time for execution of NSGAII algorithm are provided in **Table 3**.

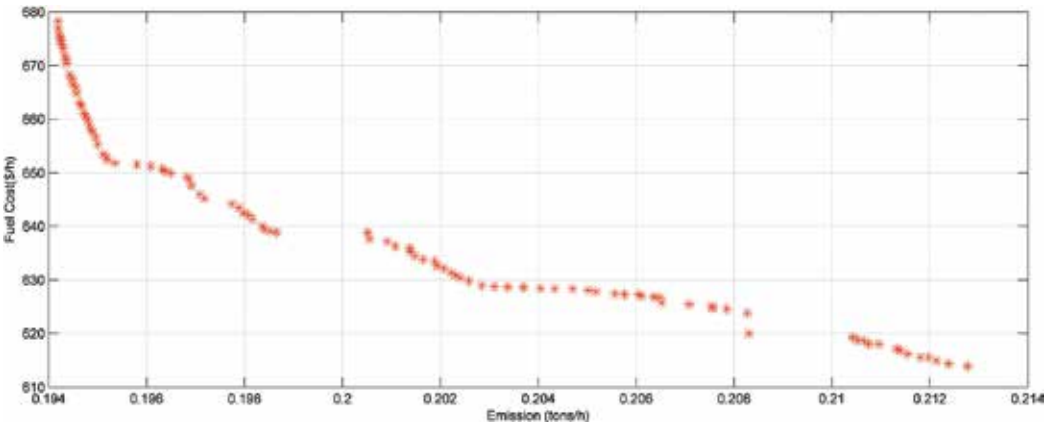

Figure 3. Pareto-optimal curve for IEEE 30 bus system obtained using Hybrid NSGAII and MOPSO Algorithm.

The parameter settings for MOPSO is obtained using trial and error is as follows: $M = 2$; Maximum number of iteration $MaxIt = 500$; Population Size $nPop = 250$; Repository size $nRep = 100$; Inertia Weight $w = 0.5$; Inertia Weight damping rate $wdamp = 0.99$; Personal learning coefficient $c1 = 1$; Global learning coefficient $c2 = 2$; Number of grids per dimension $nGrid = 10$; Inflation Rate $alpha = 0.1$, leader selection pressure $beta = 2$, Deletion selection pressure $gamma = 2$; Mutation rate $mu = 0.1$. The extreme points of the Pareto front and time for execution of MOPSO algorithm are provided in **Table 3**. We can observe from **Figure 2** and **Table 3** that there are difficulties in MOPSO algorithm in obtaining well spread Pareto front and also very slow convergence to the Pareto front when compared to NSGA II. This can be improved if the proposed hybrid approach is used to solve the CEED problem.

The Parameter setting for the hybrid algorithm is same as those given above expect for the settings provided here Population Size $nPop = 200$; Maximum number of iteration $MaxIt = 50$; Repository size $nRep = 20$. The extreme points of the Pareto front and time for execution of the proposed NSGAII-MOPSO hybrid algorithm are provided in **Table 3**. From **Table 3** it is clear that the extreme points found by the hybrid algorithm are better than NSGA II and MOPSO

Method	Pg_1	Pg_2	Pg_3	Pg_4	Pg_5	Pg_6	Pl	Fuel Cost ($/h)	Emission (Tons/h)	Time Taken (s)
NSGA II	0.0649	0.3866	0.6851	0.7999	0.5399	0.3886	0.03126	**616.426**	0.2121	**367**
	0.4070	0.4528	0.5416	0.4198	0.5365	0.5087	0.03279	677.941	**0.1942**	
MOPSO	0.0626	0.4106	0.6885	0.7994	0.5472	0.3564	0.03090	**618.211**	0.2125	1507
	0.4412	0.4574	0.5501	0.3821	0.5523	0.4832	0.03242	678.702	**0.1943**	
Hybrid NSGAII-MOPSO	0.0500	0.3893	0.6861	0.8001	0.5490	0.3911	0.03178	**613.85**	0.2127	662
	0.4109	0.4563	0.5429	0.4002	0.5435	0.5128	0.03279	678.30	**0.1942**	

Table 3. Comparison of extreme points (shown in bold) and time taken for convergence using NSGAII, MOPSO and Hybrid NSGA II-MOPSO for IEEE30 bus system with valve point loading.

http://dx.doi.org/10.5772/intechopen.72807

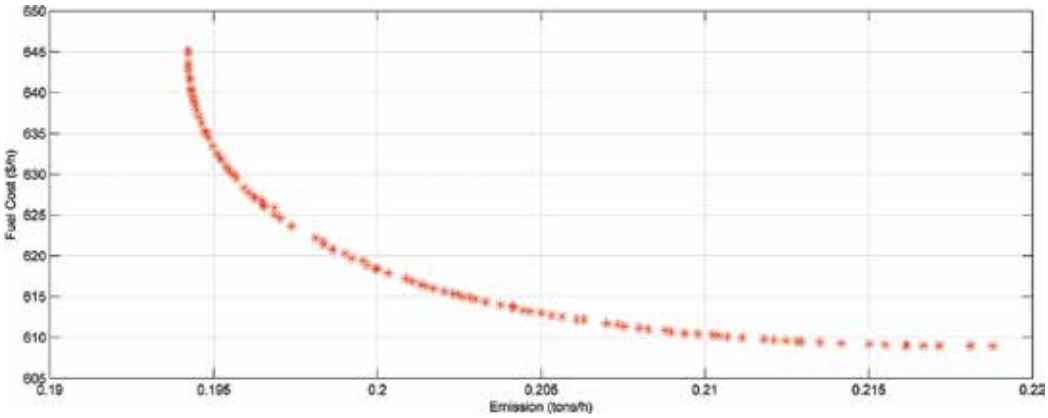

Figure 4. Pareto-optimal curve for IEEE 30 bus system without valve point effect obtained using hybrid NSGAII and MOPSO algorithm.

algorithm. Even though the time of execution of the Hybrid algorithm is slower than NSGA II it is able to find well spread Pareto front compared to NSGA II. The hybrid algorithm is far superior to MOPSO in terms of converge speed and also in finding well spread Pareto-optimal front.

In case II the valve point effect is neglected from the fuel cost curve and is solved using the proposed hybrid approach using the same parameters. The Pareto front obtained is shown in **Figure 4** and is a continuous curve when compared to the Pareto front shown in **Figure 3**. In **Figure 3** the Pareto front is discontinuous due to the effect of the Valve point loading in the cost curve. Both these case studies indicate that the hybrid approach is effective to solve the CEED problem.

7. Conclusion

In this chapter, a hybrid multi-objective optimization algorithm based on NSGA II and MOPSO have been proposed to solve the highly nonlinear, highly constrained combined economic emission dispatch problem. At any stage of the algorithm, only feasible solution is created because of the incorporation of the proposed constraint handling mechanism. During every iteration of the hybrid algorithm new population is created and NSGA II is applied on best performing individuals whereas MOPSO is applied on the lower ranked individuals to strengthen the exploration and exploitation capability of the algorithm. This hybrid approach is tested on an IEEE 30 bus system. The results obtained shows that the hybrid approach is efficient for solving CEED problem and is also able to quickly converge to a better Pareto-optimal front when compared to MOPSO algorithm. The result obtained by the hybrid approach also demonstrates it is able to yield a wide spread of solutions and convergence to true Pareto-optimal fronts.

Acknowledgements

The author gratefully acknowledges the authorities of Royal Commission for Jubail and Yanbu and the authorities of Jubail Industrial College for the facilities offered to carry out the work. This work is part of the JRICH (Jubail Research and Innovation Cluster Hub) initiative of the Jubail Industrial College.

Author details

Arunachalam Sundaram

Address all correspondence to: mailtoarunachalam@gmail.com; sundaram_a@jic.edu.sa

Department of Electrical and Electronics Engineering Technology, Jubail Industrial College, Kingdom of Saudi Arabia

References

[1] Wood AJ, Woolenberg BF. Power Generation, Operation and Control. 2nd ed. New York: John Wiley and Sons; 1996

[2] Le KD, Golden JL, Stansberry CJ, Vice RL, Wood JT, Ballance J, Cauley GW. Potential impacts of clean air regulations on system operations. IEEE Transactions on Power Systems. 1995;**10**(2):647-656. DOI: 10.1109/59.387899

[3] Deb K. Multi-Objective Optimization Using Evolutionary Algorithms. New York: Wiley; 2001. p. 518. ISBN: 978-0-471-87339-6

[4] Abido M. A niched Pareto genetic algorithm for multi-objective environmental/economic dispatch. Electric Power and Energy System. 2003;**25**(2):97-105. DOI: 10.1016/S0142-0615(02)00027-3

[5] Basu M. Dynamic economic emission dispatch using non dominated sorting genetic algorithm-II. International Journal of Electric Power and Energy Systems. 2008;**30**(2):140-149. DOI: 10.1016/j.ijepes.2007.06.009

[6] Rajasomashekar S, Aravindhababu P. Biogeography based optimization technique for best compromise solution of economic emission dispatch. Swarm and Evolutionary Computation. 2012;**7**:47-57. DOI: 10.1016/j.swevo.2012.06.001

[7] Zou D, Li S, Li Z, Kong X. A new global Particle Swarm Optimization for the economic emission dispatch with or without transmission losses. Energy Conversion and Management. 2017;**139**:45-70. DOI: 10.1016/j.swevo.2012.06.001

[8] Panigrahi BK, Ravikumar Pandi V, Sanjay D, Das S. Multiobjective fuzzy dominance based bacterial foraging algorithm to solve economic emission dispatch problem. Energy. 2010;**35**(12):4761-4770. DOI: 10.1016/j.energy.2010.09.014

[9] Silva MAC, Klein CE, Mariani VC, Coelho LS. Multiobjective scatter scearch approach with new combination scheme aplied to solve environmental/economic dispatch problem. Energy. 2013;**53**(5):14-21. DOI: 10.1016/j.energy.2013.02.045

[10] Roy PK, Bhui S. Multi-objective quasi oppositional teaching learning based optimization for economic emission load dispatch problem. Electrical Power and Energy system. 2013;**53**(4):937-948. DOI: 10.1016/j.ijepes.2013.06.015

[11] Abou El Ela AA, Abido MA, Spea SR. Differntial evolution algorithm for emission constrained economic power dispatch problem. Electric Power Systems Research. 2010;**80**(10):1286-1292. DOI: 10.1016/j.epsr.2010.04.011

[12] Sivasubramani S, Swarup KS. Environemntal/economic dispatch using multi-objective harmony search algorithm. Electric Power Systems Research. 2011;**81**(9):1778-1785. DOI: 10.1016/j.epsr.2011.04.007

[13] Arunachalam S, Saranya R, Sangeetha N. Hybrid artificial bee colony algorithm and simulated annealing algorithm for combined economic and emission dispatch including valve point effect. LNCS. 2013;**8297**(Part I):354-365. DOI: 10.1007/978-3-319-03753-0_32

[14] Arunachalam S, Agnes Bhomila T, Ramesh Babu M. Hybrid Particle Swarm Optimization algorithm and firefly algorithm based combined economic and emission dispatch including valve point effect. LNCS. 2015;**8947**:647-660. DOI: 10.1007/978-3-319-20294-5_56

[15] Palanichamy C, Babu NS. Analytical solution for combined economic and emission dispatch. Electric Power Systems Research. 2008;**78**(7):1129-1139. DOI: 10.1016/j.epsr.2007.09.005

[16] Zhiyong L, Weiyou W, Yanyan Y, Li Z. PS-ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems. Expert System with Applications. 2015;**42**:8881-8895. DOI: 10.1016/j.eswa.2015.07.043

[17] Gherbi YA, Bouzeboudja H, Gherbi FZ. The combined economic environmental dispatch using new hybrid meta heuristics. Energy. 2016;**115**:468-477. DOI: 10.1016/j.energy.2016.08.079

[18] Gong D-W, Zhang Y, Qi C-L. Environmental/economic power dispatch using hybrid multi-objective optimization algorithm. Electric Power and Energy System. 2010;**32**:607-614. DOI: 10.1016/j.ijepes.2009.11.017

[19] Agarwal A, Nanavati N. Association rule mining using hybrid GA-PSO for multi-objective optimisation. In: IEEE International Conference on Computational Intelligence and Computing Research; Dec 2016; Agni College of Technology. Chennai: IEEE; 2017. DOI: 10.1109/ICCIC.2016.7919571

[20] Gandelli A, Grimaccia F, Mussetta M, Pirinoli P, Zich RE. Development and validation of different hybridization strategies between GA and PSO. In: IEEE Congress on Evolutionary Computation; 25–28 September 2007; Singapore. IEEE; 2008. DOI: 10.1109/CEC.2007.4424823

[21] Janga Reddy M, Nagesh Kumar D. An efficient multi-objective optimization algorithm based on swarm intelligence for engineering design. Engineering Optimization. 2007;**39**(1):49-68. DOI: 10.1080/03052150600930493

[22] Kalyanmoy D, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multi-objective genetic algorithm: NSGA II. IEEE Transaction on Evolutionary Computations. 2002;**6**(2):182-197. DOI: 10.1109/4235.996017

[23] Coello Coello CA, Lechuga MS. MOPSO: A proposal for multiple objective Particle Swarm Optimization. In: Proceedings of the 2002 Congress on Evolutionary Computation; 12–17 May 2002; 07 August 2002. Honolulu: IEEE; 2002. DOI: 10.1109/CEC.2002.1004388

[24] Chaing C-L. Improved genetic algorithm for power economic dispatch of units with valve-point effects and multiple fuels. IEEE Transactions on Power System. 2005;**20**(4):1690-1699. DOI: 10.1109/TPWRS.2005.857924

[25] Srinivas N, Ded K. Multi-objective optimization using non-dominated sorting in genetic algorithm. Evolutionary Computation. 1994;**2**(3):221-248. DOI: 10.1162/evco.1994.2.3.221

[26] Hemamalini S, Sishaj PS. Emission constrained economic dispatch with valve point effect using Particle Swarm Optimization. In: TENCON; 18–21 November 2008; Hyderabad. IEEE; 2009. DOI: 10.1109/TENCON.2008.4766473

Edited by Pakize Erdoğmuş

This book is intended to gather recent studies on particle swarm optimization (PSO). In this book, readers can find the recent theoretical developments and applications on PSO algorithm. From the theoretical aspect, PSO has preserved its popularity because of the fast convergence rate, and a lot of hybrid algorithms have recently been developed in order to increase the performance of the algorithm. At the same time, PSO has also been used to solve different kinds of engineering optimization problems. In this book, a reader can find engineering applications of PSO, such as environmental economic dispatch and grid computing.

Published in London, UK

IntechOpen

ISBN 978-1-83881-430-4