Optimization Techniques: An Overview

Shubhabrata Datta, Sandipan Roy and J. Paulo Davim

Abstract There are several types of optimization methods having their own advantages and disadvantages. In recent times, metaheuristic optimization techniques are gaining attention and being applied to various industrial applications. In this chapter, a brief description of classes of optimization techniques is followed by an elaboration of the most popular optimization techniques, which are getting substantial uses in the industries. The above techniques include evolutionary algorithms, swarm intelligence techniques and simulated annealing.

Keywords Optimization methods · Classification · Evolutionary · Computation Swarm optimization · Ant colony optimization · Simulated annealing

1 Introduction

Optimization is a method to search the best solution in a given circumstance. Engineers need to decide on several technological and managerial issues for design, manufacturing, construction, as well as maintenance of systems. Minimization of efforts with maximization of the desired output is the most crucial aspect of all search decisions. Thus, efforts required to be given and/or to obtain the desired output is a function of certain variables and can be optimized to achieve the optimum level of that function [1, 2].

S. Datta (✉) · S. Roy
Department of Mechanical Engineering, SRM Institute of Science and Technology, Kattankulathur, Chennai 603203, India
e-mail: shubhabrata.p@ktr.srmuniv.ac.in

J. P. Davim
Department of Mechanical Engineering, University of Aveiro, Campus Santiago, 3810-193 Aveiro, Portugal

S. Datta and J. P. Davim (eds.), *Optimization in Industry*, Management and Industrial Engineering, https://doi.org/10.1007/978-3-030-01641-8_1

Different applications of optimizations in the field of engineering are as follows:

- Optimum design of civil engineering structures.
- Minimum-weight design of structures for earthquake, and other types of random loading.
- Design of minimum cost materials management methods.
- Design of maximum efficiency pumps, turbines, and equipment for heat transfer.
- Design optimization of electrical machinery and networks.
- Planning, scheduling, and controlling of manufacturing processes.
- Design of materials with improved performance.
- Decreasing the manufacturing cost, etc.

During the design process, an engineering system is defined by a set of variables on which the performance of the system depends. As a common aspect, some of the variables are fixed at the beginning as preassigned parameters. The other parameters are analyzed during the design optimization process, which are called design or decision variables [1]. In many cases, these design variables had to satisfied pre-specified requirements, hence, they cannot be chosen arbitrarily. To produce an acceptable design, these restrictions have to be satisfied and they are called optimization constraints. Among the constraints, behavior or functional constraints restrict the behavior of the system. Those constraints which indicate restrictions on design variables are called geometric or side constraints. In case of application to the manufacturing industry the plant constraints, describing the limits of the plant capacity, need to be considered during optimization. Any computed design usually leads to situations, which provide feasible and acceptable path toward the solution of the problem. Naturally, several other designs of the same problem could be computed with superior or inferior performances. To fulfill the purpose of optimum outcome, the best among all the acceptable designs available needs to be chosen. Thus, a decisive factor must be selected which could compare among the several acceptable designs and the optimum design could be chosen.

This decisive factor for optimizing the design is called the objective function. The objective function needs to be formulated depending on the inherent mechanism of the problem [1]. For example, in the case of structural design problems weight of automobiles or aircraft and cost in civil structures are common objective functions for minimization. Similarly, mechanical efficiency in mechanical engineering systems design is an objective function for maximization. An optimization algorithm starts with random initial solutions and improves in iterative methods. After certain iterations, it converges to the optimum solutions for the problem [3, 4]. The algorithm makes the system attracted towards the optimum solution from the solutions existing at different states, as a self-organizing system. Such an iterative system can be developed using some mathematically described equations or rules [5].

2 Classification of Optimization Methods

Optimization techniques can be classified from various aspects. It can be divided based on the presence of constraints. If the optimization problems have one or more constrained, it is called constraints optimization, and otherwise defined as an unconstrained optimization problem. Optimization can also be classified according to the number of objectives. In a single-objective optimization, either in search of maximum or minimum optimum solution, it leads to a single optimum solution. If there is more than one objective having conflict between them, the situation calls for multiobjective optimization. Here, a set of optimum non-dominated solutions evolve, which is represented in the form of Pareto front [6]. If the number of objectives is more than three, it is also called a many-objective optimization. The optimization problems can also be classified based on the nature of the objective functions as well as the constraints. The problem is called a linear or nonlinear problem based on the objective function and the constraints being linear or nonlinear. If the objective function is polynomial, it is called a geometric problem. If the constraints are integrated with the objective function and cannot be separated, then the problem is called a non-separable programming problem. When it comes to time constraints, optimization can be distinguished into two main types, online and offline optimization. If the job needs to be solved within a few seconds or milliseconds, it is called online optimization [6]. In such cases, the focus of the algorithm is speed. Robot localization, job schedule updating, transport process, and search engine are an example of such optimization.

These examples indicate that the online optimization has to be carried out in a repetitive manner to cater different orders arriving continuously to the system without significant weighting. In cases where time is not important as users are ready to wait for the optimal or close-to-optimal results, offline optimization is acceptable [6]. In case of offline optimization, the process is carried out only once. Here the optimization strategies are more important and have to be decided before the starting of the process. Again, deterministic and heuristic algorithms are two classes of the optimization algorithm.

In the case of deterministic algorithms, an obvious relation between the attributes of the system exists. It becomes hard to solve a problem deterministically when the relation between the system parameter and the objective/fitness of the system are complicated or obscure.

A heuristic algorithm [7, 8] gathers information about the system and fitness of random solution is tested and decision is taken for generating the next solution. Thus, these methods are dependent on the nature of the problem. On the other hand, a metaheuristic technique combines the heuristics and the objective function efficiently without depending on the structure of the problem [9, 10]. In this way, a metaheuristic algorithm makes the heuristic methods applicable to a wide range of problems.

High-quality solution for combinatorial optimization problems can be found by metaheuristics in a reasonable time [11].

A single-run algorithm like constructive methods or iterative improvement generate a limited number of solutions and even stop at local optima [12, 13]. If a function is difficult from the mathematical point of view when the context is discontinuous, not differentiable, or having multiple optima, a deterministic or classical optimization algorithm may converge to solutions in the local optima or from a small part of the search space. Thus, the problem with classical optimization algorithms is that generally, it is impossible to find whether the generated solution is situated in a local or global optima space. Similarly, it can also be stated that whether the provided solution is the result of the whole search space or only a part of it. This issue is more important for a multimodal problem, where multiple optima exist [14–17].

The concepts of metaheuristic optimization techniques are getting more and more accepted in real-life applications due to their robustness and capability to deal with complex design problems as discussed above. The metaheuristic techniques can be broadly divided into three groups. The first is the evolutionary computation group, consisting of genetic algorithm, genetic programming, evolutionary strategy, differential evolution, etc. The second group can be named swarm intelligence group consisting of particle swarm optimization, ant colony optimization, etc. The techniques within these two groups can also be termed as bioinspired optimization techniques. The third group consists of physical-process-inspired optimization techniques, and simulated annealing is the most common optimization tool in this group. The groups are shown in Fig. 1. Later in this chapter, some of these techniques will be discussed.

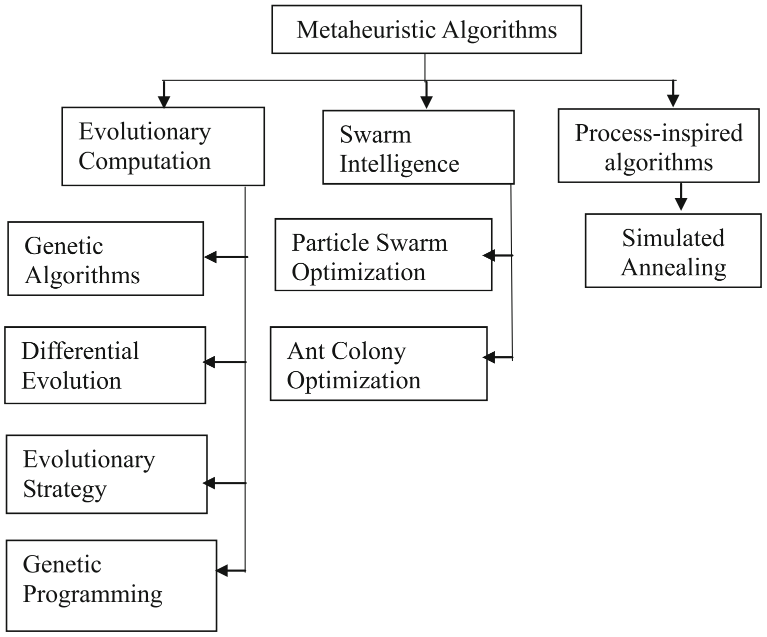

Fig. 1 Major classes of metaheuristic global optimization algorithms with important variants

3 Evolutionary Computation

Evolutionary computation (EC) uses principles inspired by the natural evolution of species and being used nowadays for solving many complex engineering problems. Four main types of this genre of algorithm are genetic algorithms [18], evolution strategies [19, 20], differential evolution [21], and genetic programming [22]. The common part of these four algorithms is that all four algorithms are populations based and use operators inspired by the concept of natural evolution. The selection operator chooses the individuals who will survive and create the next generation. If the objective function provided better fitness of an individual, then the probability of selection of the individuals is higher. The selected individuals are combined to form offspring through an operator called crossover, which merges parts of two selected individuals to generate in new individuals. The mutation operator introduces random modification within one individual to incorporate diversity of the solutions. These algorithms differ in representations of the individuals and the method of implementing the operators. For example, the mutation operator is more emphasized in the case of evolutionary strategies and genetic programming algorithm.

Compared to other methods, the most significant advantage of EC is that little knowledge about the system is required to solve the problem, and thus these can be applied to a wide range of optimization problems. Even if the fitness function is not continuous, not differentiable and highly complex then also EC can solve the problem efficiently [23]. Additional information or knowledge about the system can also be incorporated in the optimization framework as constraints to improve the performance of the algorithm and to achieve better solutions.

As the different types of EC algorithms are available, users can easily choose the methods suited for a given problem depending on the structured of the system to be optimized. These methods can easily be used for multi-objective optimization problems [24]. EC provides a described heuristic estimation of optimum solution through a process based on certain operators [25–30]. The schema of the EC Algorithm is given in the form of a flowchart in Fig. 2.

All the evolutionary computing algorithms follow the same general methods with certain variations. The structure of a solution varied from one algorithm to other. In the case of Genetic Algorithms (GA), the solutions are represented in the form of a stream. In the case of Evolution Strategies (ES), it is a real-valued vector, and in the case of Genetic Programming (GP) it is represented as a tree. This difference in the structures of candidate solution makes the different types of EC algorithms applicable for different types of optimization problems. Genetic algorithms (GA) are natural selection and natural genetics inspired optimization algorithms. It can be used for a wide range of problems with high complexity in the objective function. To solve practical engineering optimization problems, the idea of using a population of solutions was evolved during the 1950 and 1960s. John Holland of the University of Michigan initiated the idea of sexual reproduction of solutions to EA [31]. This initiation of the new method of the optimization considered makes Holland considered to be the father of Evolutionary Algorithm. His book, written in 1975

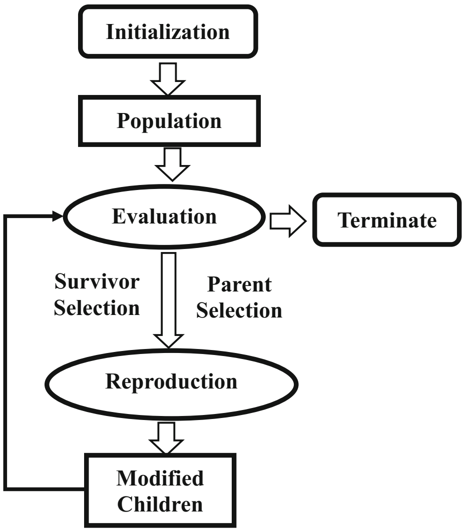

Fig. 2 The scheme of an evolutionary computing

[31] particularly gives an idea about his creative thinker approach. De Jong, in one of his papers entitled Genetic Algorithms emphasized that GA is just NOT Function Optimizers [32] and has more potential than a mere robust method for optimizing and engineering system.

Though there are certain differences in EC algorithms, the basic principles of assessing the performance of the solution and then keeping the better performing solutions for further processing and make the worst performers perish remains the same for all cases [33]. Population-based incremental learning (PBIL) [34], a type of EC algorithm has similarities to ant colony optimization which is discussed later. PBIL has a generating vector which is nothing but a vector and probabilities. Binary strings representing the solutions generated randomly has the ith bit having a probability value corresponding to the generating vector. When the solutions in the population are evaluated through the fitness function the probability values in the generating vector is updated is depending on the quality of the performance of the solutions. In case of ant colony optimization, pheromone trail values are similarly updated depending on the performance.

4 Swarm Optimization

In 1995 by Eberhart and Kennedy developed Particle Swarm Optimization (PSO), which is an optimization method inspired by the behaviors of birds flocking or fish schooling [35]. Later several variations of PSO have emerged for improving the convergence speed and solution quality. Basic PSO is found to be more appropriate for processing simple as well as static optimization problems. The PSO imitate behavior of such animal societies those find their food without the existence of any leader in the group or swarm. In such cases, the members of the group try to follow other members of the group depending on the closeness of the member from the source of food. This can only be possible due to the presence of communication system among the members of the group. The member who is in a better condition should have the facility to inform the other members of the flocks regarding his present position which can be followed by the other members. This situation is repeated until the food source could be reached. This basic concept of animal behavior is used by PSO for finding optimal solutions. Each particle of the swarm of a particle in particle optimization represents potential solutions.

To improve the convergence speed and solution quality in PSO several variations have been developed, viz., inertia weight, velocity clamping, synchronous and asynchronous updates, constriction coefficient [36]. PSO algorithm [5] adjusts the trajectories of the agents during its search for the optimum solutions in the search space. Each agent or particle compares its current position with its own best location during its random movement. As it reaches a location better than its previous best location, it updates the best location. In this way, the particles search the global best location during the random search. Ant colony optimization (ACO) is another global optimization technique, which falls within the swarm optimization group. Ant colonies are generally formed by social insect societies. Due to the highly structured social organization, ant colonies can accomplish complex tasks. The behavior of real ant acts as the source of the information for this novel algorithm, where the principle of self-organizing of ants in a highly coordinated manner is exploited. In case of SEO populations of artificial ants collaborate between themselves to solve optimization problems. Different ant algorithms have evolved depending on various aspects of ant behavior which include division of labor, foraging, cooperative transport, and brood sorting. In all the above activities, ants' coordinates among themselves through indirect communication methods called stigmergy. In case of foraging a chemical named pheromone, a deposited on the ground by the ants to inform the other ants about its path of movements and directing them to follow the same path. This stigmergic communication system has been proved by the biologists as the methods to achieve the self-organization. One of the most flourishing examples of several ant algorithms is ACO. The communication through pheromone which detected by the antenna of ants make the ants follows the path where a larger amount of pheromone is present. This behavior pursuit by real ants can be considered as a heuristic method. The initial explorations of the ants in the neighboring area occur in a complete random manner without the presence of pheromone. But the addition of pheromone incorporates a certain amount of control over the random search. During the random travel deposition by the pheromone by the ants provide some feedback to the other ants and search methods becomes direction control to a certain extent. As pheromone is evaporating in nature an unused path disappeared with time [37].

Dorigo and Gambardela [38] proposed an improved algorithm, where the three main differences with the previous and system are transition rule, local updating, global updating. The transition rule is made to generate a relation between the explorations and priority of the problem. The local updating methods update the pheromone

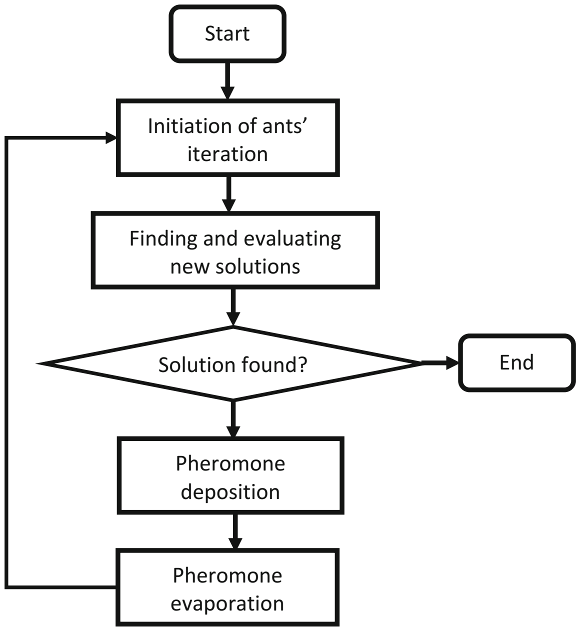
Start
Initiation of ants' iteration
Finding and evaluating new solutions
Solution found?
End
Pheromone deposition
Pheromone evaporation

Fig. 3 Structure of ACO algorithm

deposition of the ants during the local search whereas global updating is done for the ants with superior position. Figure 3 shows a general scheme of the Ant Colony Swarm algorithm.

5 Simulated Annealing

Simulated Annealing (SA) mimics a physical process called 'annealing', which has a close relation with thermodynamics, and a method to control the microstructure and properties of metals and alloys through a process of heating and cooling [39]. In this process, the temperature is reduced slowly so that the material can achieve an internal structure or arrangements of atoms in the lowest energy configuration or equilibrium condition. In simulated annealing the objective function replaces the energy of the material being annealed. In case of a minimization problem, lower solution will have lower energy. The optimization method is made of a random move for hill-climbing. Every move gets a probability as it related to the energy or the function value using a parameter similar to Boltzmann constant. Thus, the probability value has an analogy of temperature in thermodynamics and the quality of the solution can be compared with the energy of the system. At higher probability, more uphill moves are possible. The overall probability starts high and is gradually decreased. In the process, optimum solution is reached by one or more moves.

6 Concluding Remarks

The metaheuristic algorithms have greater flexibility to handle problems with complex objective functions and constraints. This reason has made these techniques more applicable for the industrial situations. In such complex real-life situations, these techniques are being used quite successfully to solve the industrial problems related to design, quality control, productivity, and cost.

References

1. Rao, S. S. (2009). *Engineering Optimization Theory and Practice* (4th ed.). Copyright © 2009.
2. Beightler, C. S, Phillips, D. T., & Wilde, D. J. (1979). *Foundations of optimization* (2nd ed.). Englewood Cliffs, NJ: Prentice Hall.
3. Koziel, S., & Yang, X. S. (2011). *Computational optimization, methods and algorithms*. Germany: Springer.
4. Yang, X. S. (2010). *Engineering optimization: an introduction with metaheuristic applications*. Wiley.
5. Yang, X.-S. (2014). *School of science and technology*. London: Middlesex University London. Copyright © 2014 Elsevier Inc. ISBN 978-0-12-416743-8.
6. Weise, T. (2009) *Global optimization algorithms—theory and application*, Version: June 26, 2009.
7. Michalewicz, Z., & Fogel, D. B. (2004). *How to solve it: Modern heuristics*. Springer, second, revised and extended edition, December 2004. ISBN: 978-3-54022-494-5.
8. Rayward-Smith, V. J., Osman, I. H., Reeves, C. R., & Smith, G. D. (Eds.). *Modern heuristic search methods*. Wiley, December 1996. ISBN: 978-0-47196-280-9.
9. Glover, F., & Kochenberger, G. A. (Eds.). (2003). Handbook of Metaheuristics, volume 57 of International Series in Operations Research & Management Science. Kluwer Academic Publishers/Springer, New York, USA. ISBN: 978-1-40207-263-5, 978-0-30648-056-0, 0-3064-8056-5, 1-4020-7263-5. https://doi.org/10.1007/b101874. Series Editor Frederick S. Hillier.
10. Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, *35*(3):268–308. ISSN: 0360-0300. CODEN: CMSVAN. http://iridia.ulb.ac.be/~meta/newsite/downloads/ACSUR-blum-roli.pdf.
11. Dorigo, M., & Stützle, T. *Ant colony optimization, a bradford book*. London, England, MA: The MIT Press Cambridge. ISBN 0-262-04219-3.
12. Johnson, D. S., & McGeoch, L. A. (1997). The travelling salesman problem: A case study in local optimization. In E. H. L. Aarts & J. K. Lenstra (eds.), *Local search in combinatorial optimization* (pp. 215–310). Chichester, UK: Wiley.
13. Schreiber, G. R., & Martin, O. C. (1999). Cut size statistics of graph bisection heuristics. *SIAM Journal on Optimization, 10*(1), 231–251.
14. Shekel, J. (1971). Test functions for multimodal search techniques. In *Proceedings of the Fifth Annual Princeton Conference on Information Science and Systems* (pp. 354–359). Princeton, NJ, USA: Princeton University Press.
15. Žilinskas, A. (1978). Algorithm as 133: Optimization of one-dimensional multimodal functions. *Applied Statistics*, *27*(3), 367–375. ISSN: 00359254. https://doi.org/10.2307/2347182.
16. Ursem, R. K. (2003). Models for evolutionary algorithms and their applications in system identification and control optimization. Ph.D. thesis, Department of Computer Science, University of Aarhus, Denmark, April 1, 2003. Advisors: T. Krink & B. H. Mayoh. http://www.daimi.au.dk/~ursem/publications/RKU_thesis_2003.pdf and http://citeseer.ist.psu.edu/572321.html.

17. Schaffer, J. D., Eshelman, L. J., & Offutt, D. (1990). Spurious correlations and premature convergence in genetic algorithms. In *Proceedings of the First Workshop on Foundations of Genetic Algorithms (FOGA)*, pp. 102–112. In proceedings (1924).
18. Goldberg, D. E. (1989). *Genetic algorithms in search optimization and machine learning*. Addison-Wesley: Reading, MA.
19. Rechenberg, I. (1973). *Evolutions strategie—Optimierung technischer Systemenach Prinzipien der biologischen Information*. Freiburg, Germany: Fromman Verlag.
20. Schwefel, H.-P. (1981). *Numerical optimization of computer models*. Chichester, UK: Wiley.
21. Price, Kenneth, Storn, Rainer M., & Lampinen, Jouni A. (2005). *differential evolution—a practical approach to global optimization*. Berlin, Heidelberg: Springer.
22. Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial intelligence through simulated evolution*. New York: Wiley.
23. F. Streichert, *Introduction to evolutionary algorithms,* presented at the Frankfurt MathFinance Workshop, April 2–4, 2002.
24. Van Veldhuizen, D. A., & Lamont, G. B. (2000). Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation, 8*(2), 125–147.
25. Bäck, T. (1996). *Evolutionary algorithms in theory and practice*. Oxford University Press, New York. A book giving a formal treatment of evolutionary programming, evolution strategies, and genetic algorithms (no genetic programming) from a perspective of optimisation.
26. Bäck, T., & Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimisation. *Evolutionary Computation*, *1*(1), 1–23. A classical paper (with formalistic algorithm descriptions) that "unified" the field.
27. Eiben, A. E. (2002). Evolutionary computing: the most powerful problem solver in the universe? *Dutch Mathematical Archive (Nederlands Archief voor Wiskunde)*, *5/3*(2), 126–131. A gentle introduction to evolutionary computing with details over GAs and ES. To be found at http://www.cs.vu.nl/~gusz/papers/ec-intro-naw.ps.
28. Fogel, D. B. (1995). *Evolutionary computation*. IEEE Press. A book covering evolutionary programming, evolution strategies, and genetic algorithms (no genetic programming) from a perspective of achieving machine intelligence through evolution.
29. Hillier, M. S., & Hillier, F. S. (2002). Conventional optimization techniques. Chapter 1. In R. Sarker, M. Mohammadian, & X. Yao, (eds.), *Evolutionary optimization*, (pp. 3–25). Kluwer Academic Publishers. Gives a nice overview of Operations Research techniques for optimisation, including linear-, nonlinear-, goal-, and integer programming.
30. Yao, X. (2002). Evolutionary computation: A gentle introduction. Chapter 2. In R. Sarker, M. Mohammadian, & X. Yao, (eds.), *Evolutionary optimization* (pp. 27–53). Kluwer Academic Publishers. Indeed a smooth introduction presenting all dialects and explicitly discussing EAs in relation to generate-and-test methods.
31. Holland, J. (1975). *Adaption in natural and artificial systems: An introductory analysis with applications to biology, control and artificial systems*. Ann Arbor: The University Press of Michigan Press.
32. De Jong, K. A. (1993). Genetic algorithms are NOT function optimisers. In L. D. Whitley (ed.), *Foundations of genetic algorithms 2*, Morgan Kaufinann.
33. Wright, S. (1931). Evolution in mendelian populations. *Genetics, 16,* 97–159.
34. Baluja, S., & Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. In A. Prieditis & S. Russell (Eds.), *Proceedings of the Twelfth International Conference on Machine Learning (ML-95)* (pp. 38–46). Palo Alto, CA: Morgan Kaufmann.
35. Kennedy, J., & Eberhart, R. C. (1995). Particle swam optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, Piscataway, NJ, USA (pp. 1942–1948).
36. Rini, D. P., Shamsuddin, S. M., & Yuhaniz, S. S. (2011). Particle swarm optimization: Technique, system and challenges. *International Journal of Computer Applications* (0975–8887), *14*(1).
37. Artificial Societies and Social Simulation using Ant Colony, Particle Swarm Optimization and Cultural AlgorithmsSource. (2010). In: Book P. Korosec (ed.), New Achievements in Evolutionary Computation (p. 318), February 2010, Croatia: INTECH. Downloaded from SCIYO.COM. ISBN 978-953-307-053-7.

38. Dorigo, M., & Gambardella, L. M. (1996). *Ant colony system: A cooperative learning approach to the traveling salesman problem*. Technical Report TR/IRIDIA/1996-5, IRIDIA, Université Libre de Bruxelles.
39. van Laarhoven, P. J., & Aarts, E. H. Simulated *Annealing: Theory and Applications (Mathematics and Its Applications) Hardcover*. Dordrecht, The Netherlands: Kluwer Academic Publishers.