CHAPTER

METAHEURISTIC ALGORITHMS: A COMPREHENSIVE REVIEW

10

Mohamed Abdel-Basset*, Laila Abdel-Fatah*, Arun Kumar Sangaiah†

**Faculty of Computers and Informatics, Zagazig University, Zagazig, Sharqiyah, Egypt †School of Computing Science and Engineering, Vellore Institute of Technology, Vellore, India*

10.1 INTRODUCTION

Concurrently with boosting information technology, a huge number of optimization problems arises in several fields such as engineering, bioinformatics, operation research, and geophysics, etc. Unfortunately, most optimization problems were classified as NP-hard problems which cannot be solved in a polynomial time region unless NP equal to P. Thus, only instances with small scale can be handled by using exact mathematical methods. Instead of giving up, the researchers thought to use possible workarounds (approximation methods) which can find a good enough solution in a reasonable time, these methods can be categorized to heuristics and metaheuristics. The significant difference between both is that heuristics are more problem-dependent than metaheuristics. In other words, heuristics can be efficiently applied to a specific problem meanwhile become insufficient to other problems. On the other hand, metaheuristic seems to be a generic algorithm framework or a black box optimizer that can be applied to almost all optimization problems.

Fred Glover coined the word "metaheuristic" in 1986 [1] in order to illustrate heuristic method with not problem-specific characteristic. The reason behind the robust searching mechanism of metaheuristic is the harmonization between two search schemas: exploration (diversification) and exploitation (intensification) [2]. For the first, it is responsible for searching in the best solution surrounded areas while the latter tends to invade new searching areas. In spite of the prosperity of metaheuristic in solving a given problem, it can't solve all optimization problems, and all metaheuristics have the same performance on average (no free lunch theorems (NFL) [3,4]). This was an impetus for developing an enormous number of metaheuristics for adapting with various problem types (continuous, discrete, unconstrained, multi-objective, etc.).

Many surveys on metaheuristics have been submitted from time to time, but there is no survey covered all aspects. For example, many papers focusing on one side of metaheuristics especially biological or nature-inspired such as [5–8], Khajehzadeh et al. [9] reviewed some metaheuristics, but did not discuss any variants, Boussaïd [10] proposed a good survey, but the discussed variants were not classified and Sörensen and Glover [11] did not provide examples and mainly focus on metaheuristics taxonomies. This chapter aims to cover all relevant aspects of metaheuristics and seriously directed to help new researchers take a brief overview about all existing metaheuristics, taxonomies, and variants. As well, several widely used metaheuristics with different metaphors are discussed in order to enhance their ideation.

Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications. DOI: 10.1016/B978-0-12-813314-9.00010-4

The structure of this chapter is as the following (see Fig. 10.1): Section 10.2 briefly outlines the main metaheuristics taxonomies, Section 10.3 discusses metaphor based metaheuristics examples. Besides, Section 10.4 reviews non-metaphor based metaheuristics examples. Some metaheuristics variants are explained in Section 10.5. A real-time case study is solved in Section 10.6. Finally, the limitation and new trends are discussed in Section 10.7 followed by conclusion in Section 10.8.

10.2 METAHEURISTICS TAXONOMIES

Roughly speaking, metaheuristic is considered to be an algorithmic structure that generally applied to a variety of optimization problems with only a few modifications for adapting to the given problem. Fig. 10.2 gives a clarification of how abstract metaheuristic works. Also, metaheuristics have fundamental characteristics can be summarized as:

- Metaheuristics are not for a particular problem.
- Metaheuristics are usually approximate.
- Metaheuristics scout about the search space to find "good enough" solution.
- Metaheuristics essentially can be described by abstraction level.
- Metaheuristics usually allow an easy parallel implementation.
- Metaheuristics extend from basic local search to advanced learning techniques.
- Metaheuristics may incorporate various mechanisms in order to avoid premature convergence.
- Heuristics can be employed by a metaheuristic as a domain-specific knowledge which is dominated by the upper-level strategy.
- Emerging metaheuristics use guidance memory that preserves search experience.

As mentioned before, exploration and exploitation are the main functions of metaheuristics and the key to efficient search process is the proper trade-off between exploration and exploitation. Different classifications of metaheuristics have been submitted according to the way of employing the exploration and the exploitation, and the metaphor of the search procedures. For example:

Osman [12] classified metaheuristics into local search, construction-based, and population-based metaheuristics. The former repeatedly makes small changes to one solution. The second one builds solutions from their component parts by adding one part at a time to an incomplete solution. Population-based metaheuristics combine solutions into new ones iteratively.

Gendreau and Potvin [13] divided metaheuristics into two categories: trajectory-based metaheuristics and population-based metaheuristics. A trajectory-based algorithm begins initially with a single solution and, at each iteration, the current best solution is replaced by a new one. While the population-based algorithm starts with randomly generating a population of initial solutions. Then, the initial population will be progressively enhanced through search iterations. At each iteration, newly generated best solutions substitute the whole population or a part of it. Generally, trajectory-based solutions are more exploitation oriented whereas population-based metaheuristics are more exploration oriented.

Fister et al. [5] divided all existing metaheuristics into non-nature inspired and nature inspired. Nature inspired metaheuristics were divided into the following: swarm intelligence (SI) based, bio-inspired (but not SI-based), physics/chemistry-based, and other algorithms that can not be classified

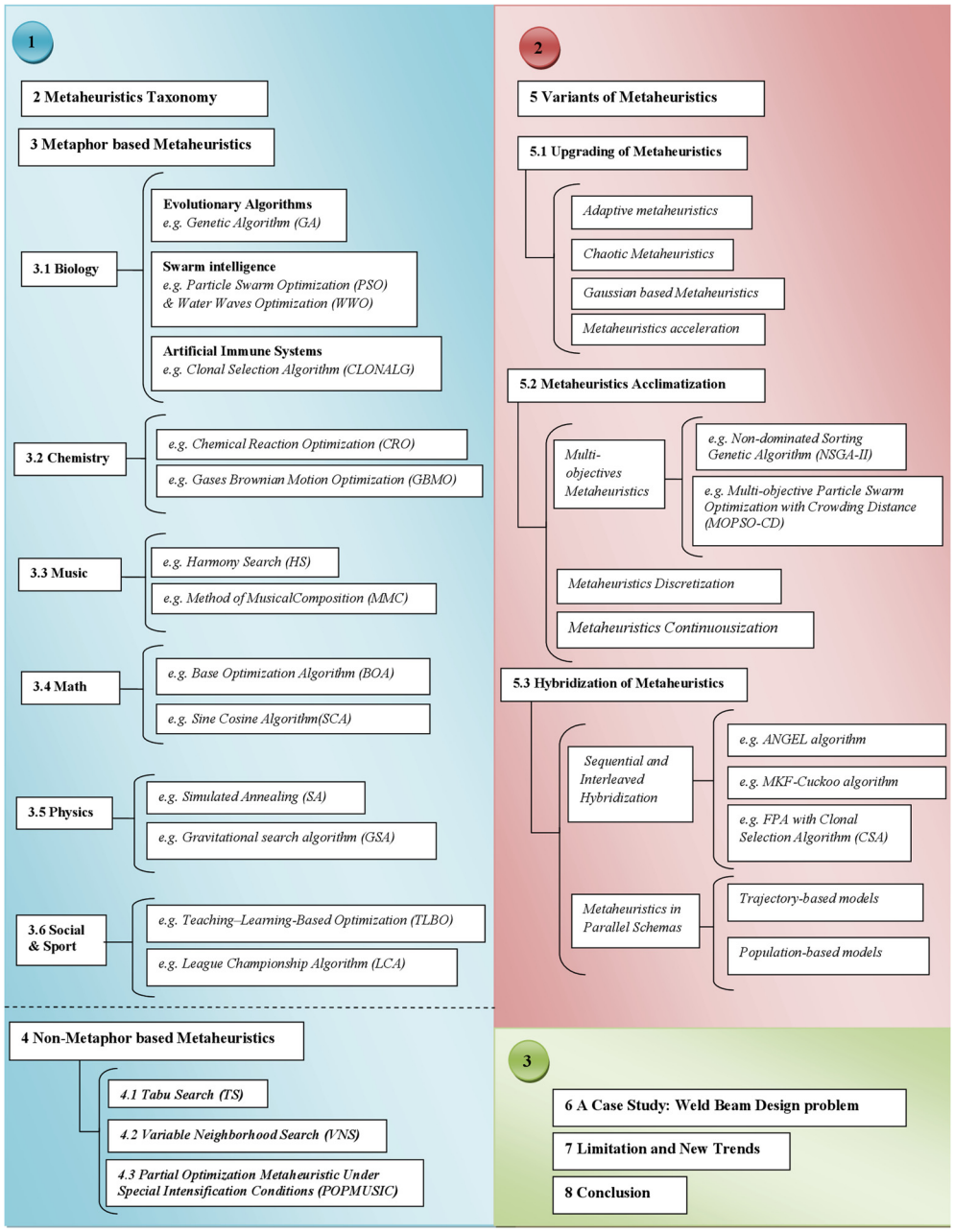

FIGURE 10.1

Chapter roadmap.

```
Create one or more initial solutions
  While (stopping criterion not satisfied) do
      If exploit then
          Create new solution by exploitation step;
      Else
          Create new solution by exploration step;
      End
      Update best found solution ;
  End
  Return best found solution;
```

FIGURE 10.2

Abstract algorithmic framework for metaheuristics.

under any previous three categories because they were inspired by diversified characteristics from different sources, such as social, emotional, etc.

Another similar classification was introduced by Akyol and Alatas [14]. Also, Binitha and Sathya [6] introduced another bio-inspired classification of metaheuristics and Ruiz-Vanoye [15] introduced a new classification of metaheuristics algorithms "not based on swarm intelligence theory" based on animals' groups: swarm, schools, flocks, and herds algorithms. Other characteristics used for the metaheuristics classification such as the manner of the objective function usage, the number of neighborhood structures, and the use of memory or not. However, the most common classification of the metaheuristics is according to "Trajectory-based & Population-based" and "Nature-inspired & Non-Nature-inspired".

In this chapter, a new classification of metaheuristics is submitted which divides metaheuristics into metaphor based and non-metaphor based metaheuristics. The metaphor based metaheuristics are algorithms that simulate natural phenomena, human behavior in modern real life or even mathematics, etc. On the other hand, non-metaphor based metaheuristics didn't use any simulation for determining their search strategy. Fig. 10.3 gives a comprehensive review of the main metaheuristic taxonomies.

10.3 METAPHOR BASED METAHEURISTICS

10.3.1 BIOLOGY BASED METAHEURISTICS

The majority of metaheuristics are based on biological evolution principles. In particular, they are concerned with simulating various biological metaphors which differ in the nature of the representation schemes (structure, components, etc.). There are three main paradigms: evolutionary, swarm, and immune systems.

Evolutionary Algorithms (EAs) simulate the biological progression of evolution at the cellular level employing selection, crossover, mutation, and reproduction operators to generate increasingly better candidate solutions (chromosomes). For evolutionary computation, there are four historical paradigms: evolutionary programming [16], evolutionary strategies [17], genetic algorithms [18], and genetic programming [19].

Swarm intelligence (SI) mimics the collective behavior of agents in a community, such as birds and insects. SI mainly depends on decentralization principle i.e. the candidate solutions are updated through

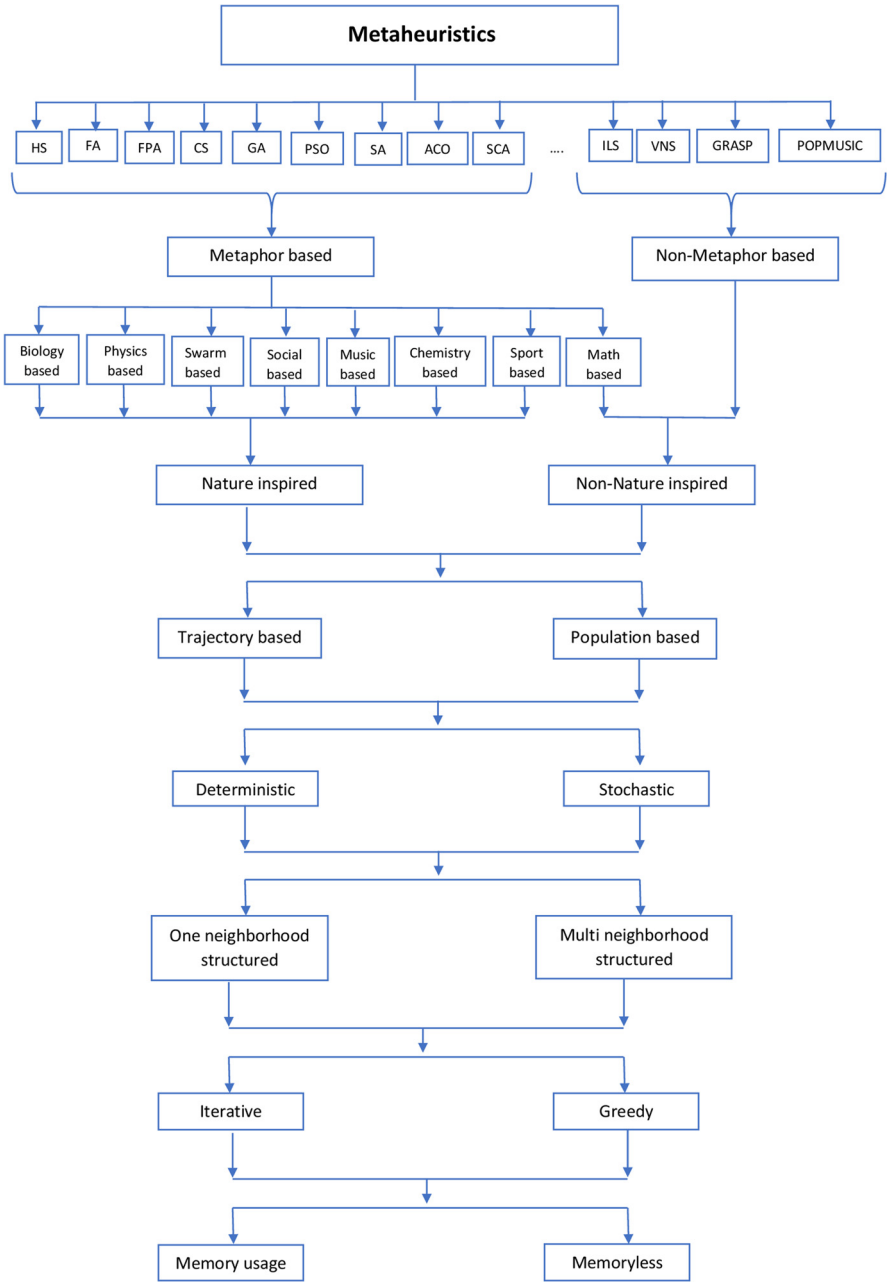

FIGURE 10.3

Metaheuristic taxonomies.

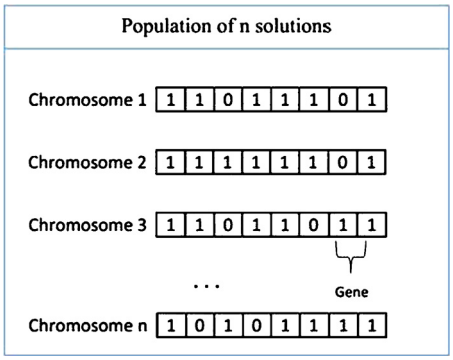

FIGURE 10.4

GA population.

the local interaction with each other and with their environment. The most popular SI algorithms are Particle Swarm Optimization (PSO) [20], Ant Colony Optimization (ACO) [21].

Artificial Immune Systems (AIS) take its inspiration from theoretical immunology and observed immune functions, principles, and models [22]. They could be seen as another algorithmic variation of Evolutionary Algorithms [23]. When applied in optimization, antibodies represent candidate solutions which iteratively evolved through repeating a cloning, mutation and selection operators. The antigen represents an objective function and good solutions are kept in a memory cell. Almost all AIS based metaheuristics depend on the clonal selection principles such as Negative Selection Algorithms [24], Clonal Selection Algorithm (CLONALG) [25], optimization version of Artificial Immune Network (opt-AINET) [26], B-Cell Algorithm [27].

It seems that the three bio-concepts are interrelated due to the similarity of the used operators (Specially selection and mutation operators). Thence, SI and AIS could be categorized as a subcategory of EAs [28,29]. However, what distinguishes the three concepts is the way they apply exploration and exploitation. Recently SI has received more popularity as it oversteps the intricate mechanisms governing the evolution of EAs and AIS. For further information [30–32]. Next, GA, PSO, CLONALG algorithms will be discussed as examples.

10.3.1.1 Genetic Algorithm (GA)

Darwin's principle "Survival of the fittest" is the starting point in introducing the mechanism of biological evolution. GA emulates the chromosomes biological evolution process by defining the following operators: selection, crossover, and mutation. Chromosomes (see Fig. 10.4) are handled as candidate solutions for a given problem and are evaluated according to their fitness. Selection of parents for breeding is an influential process for generating new solutions. There are different selection schemas such as the roulette wheel selection (RWS), the tournament selection (TOS), the linear rank selection (LRS), and the truncation selection (TRS), etc.

During crossover phase, some parts of two selected chromosomes are swapped. Chromosomes parts can be exchanged in different ways such as one, two, and uniform crossover, etc. In Mutation, some chromosomes' parts are changed randomly so as to escape from local optima. However, the best

```
Generate an initialize population randomly;
Evaluate each individual;
While (termination criterion is not satisfied) do
    Select parents;  // the best individuals are selected.
    Crossover;
    Mutation;
    Evaluate new individuals;
    Replace worst individuals by the best new ones;
End
```

FIGURE 10.5

GA pseudocode.

chromosomes can be lost when creating new chromosomes. To avoid that, the elitism is used to copy the best chromosome (or a few best chromosomes) to the new population (see Fig. 10.5). For checking the validity of GA, various analysis and studies were submitted such as [33–38].

10.3.1.2 Particle Swarm Optimization (PSO)

PSO was inspired from animals behavior in society as fish schooling or bird flocking. It can be considered as a semi-evolutionary swarm intelligence algorithm, i.e. like EA, a population of solutions is randomly sampled and evaluated to determine the best solution and repeat the same processes for a predefined number of times. Unlike an EA, each candidate solution is explicitly associated with a search process, which also has a velocity and a memory of its best positions that has so far.

Through the search procedure, candidate solutions (particles) are updated in n-dimensional space according to a position x_i and velocity v_i. The new particle position is calculated as follows:

$$v_i^{t+1} = v_i^t + \alpha\varepsilon_1[pbest_i^t - x_i^t] + \alpha\varepsilon_2[gbest^t - x_i^t] \tag{10.1}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{10.2}$$

where i is a particle index, t is a discrete time index, $pBest$ is a personal best, $gBest$ is a global best value, α and β are positive constants ($usually \approx 2$) for speeding control of the particle's flying, and ε_1 and ε_2 are two random vectors in the range $[0, 1]$ (see Fig. 10.6). For a detailed analysis of PSO performance, many papers were submitted as [39–42]. Besides, Zhang et al. [43] proposed a comprehensive survey on PSO.

10.3.1.3 Water Waves Optimization (WWO)

WWO is a novel metaheuristic proposed by Zheng [44]. The main idea of WWO is mimicking the shallow water waves theory. In WWO, each candidate solution is presented as a wave, and the search process is perceived as wave motions including propagation, breaking, and refraction. Also, the fitness of the wave $x \in X$ is inversely depending on its depth. The algorithm procedures begin with the initial generating of candidate solutions (waves) where each of them has its own height $h \in Z^+$ and length $\lambda \in R^+$. During the search process, the new wave x^{t+1} is generated using the propagation as:

$$x^{t+1}(d) = x^t(d) + \delta \bullet \lambda L(d) \tag{10.3}$$

where δ is a random number inside the range $[-1, 1]$ and $L(d)$ denotes the length of the dth dimension of the search space ($1 \leq d \leq n$). Then, the current best wave is updated and a wave breaking is applied

```
Initialize particles ;      // position and velocity
Do
   For each particle
      Calculate Data fitness value;
      If the fitness value is better than pBest
         Set pBest = current fitness value ;
      end
      If pBest is better than gBest
         Set gBest = pBest ;
      end
   end
   For each particle
      Calculate particle Velocity;
      Use gBest and Velocity to update particle position;
   end
While (termination criterion is not satisfied)
```

FIGURE 10.6

PSO pseudocode.

as:

$$x^{t+1}(d) = x^t(d) + N(0,1)\beta L(d) \tag{10.4}$$

where β is the breaking coefficient. In the case of that, the fitness of x^{t+1} is less than the fitness of x^t, it will be kept and the height will be decreased by one which simulates the energy dissipation because of inertial resistance, vortex shedding, and bottom friction. If the wave height is decreased to zero, the wave is replaced with a new randomly generated one by the refraction operator as:

$$x^{t+1}(d) \sim N\left(\frac{x^*(d) + x^t(d)}{2}, \frac{|x^*(d) - x^t(d)|}{2}\right) \tag{10.5}$$

where x^* is the current best solution and $N(\mu, \sigma)$ is the Gaussian random number with mean μ and standard deviation σ. After refraction, the height of x^{t+1} is reset to h_{max} and the length is updated (see Fig. 10.7).

WWO was tested on 30 benchmarks and the overall results of WWO outperform many well-known algorithms such as Invasive Weed Optimization (IWO) [45], Biogeography-Based Optimization (BBO) [46], Gravitational Search Algorithm (GSA) [47], Hunting Search (HUS) algorithm [48], and Bat Algorithm (BA) [49]. In addition, WWO was applied to high-speed train scheduling problem in China to show its efficiency in the real life application. For a detailed analysis of the WWO performance, Zhang and Zheng [50] made a valuable analysis of WWO convergence conditions.

10.3.1.4 Clonal Selection Algorithm (CLONALG)

Burent [51] first introduced the theory of Clonal Selection in 1959 in order to discuss the adaptive immune system basic response (lymphocytes) to antigenic stimulus. He confirms that only those cells capable of identifying an antigen will proliferate, whereas those that do not identify an antigen are selected against. CLONALG is the optimization version of Clonal Selection paradigms as the natural immune system can have multiple or contradictory goals and it has no reason to develop an optimal response. As shown before, a population of antibodies (candidate solutions) is generated randomly and

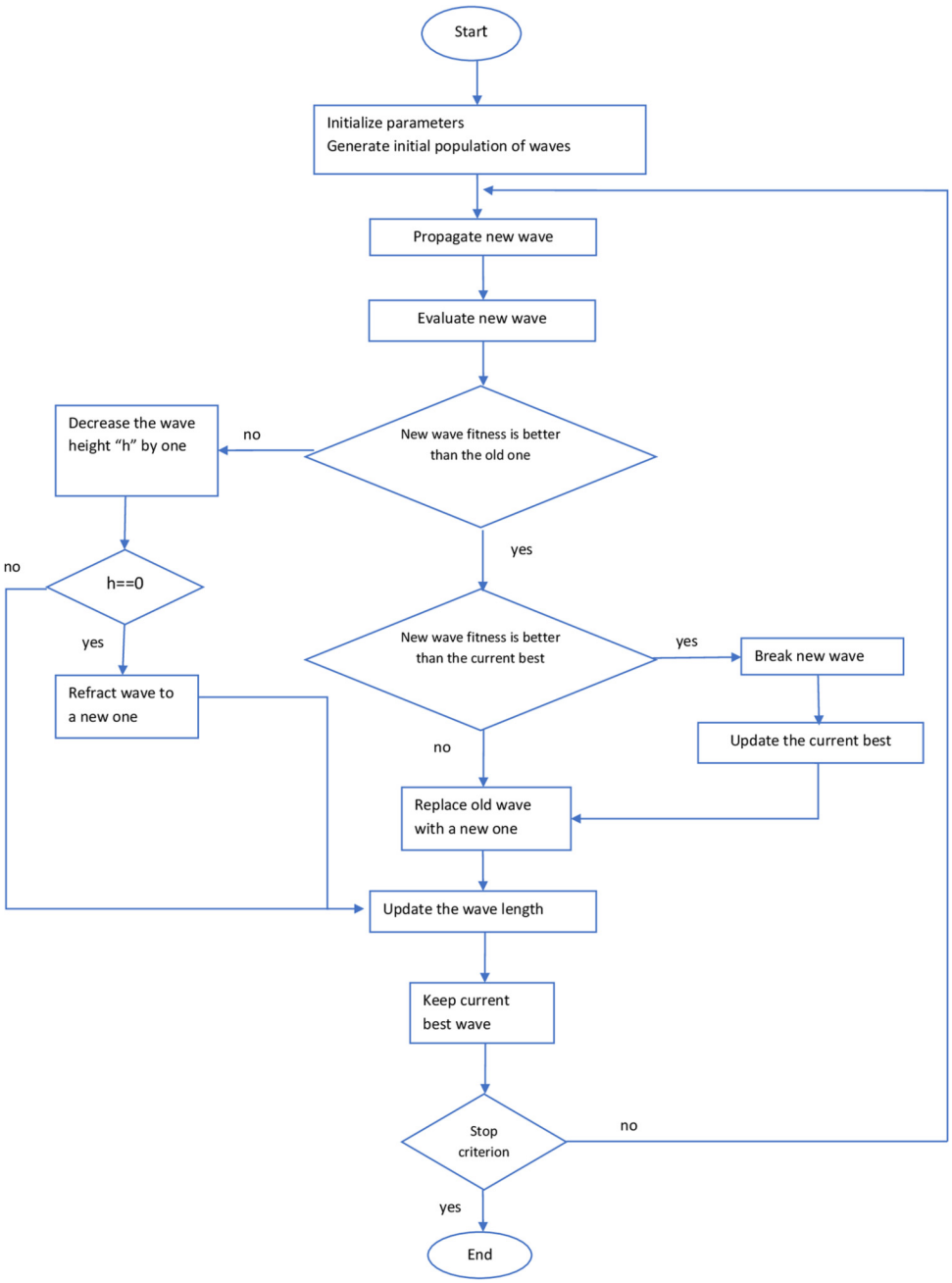

FIGURE 10.7

WWO pseudocode.

```
Generate initial population randomly;
Evaluate each solution;
While (termination criterion is not satisfied) do
        Evaluate each solution;
        Select best antibodies;
        Clone;
        Mutate;
        Keep the current best antibodies;
        Replace not cloned antibodies with new ones;
End
```

FIGURE 10.8

CLONALG pseudocode.

evaluated. Then, the higher affinities antibodies are cloned in order to generate more antibodies against the antigen. Those antibodies are not cloned are replaced by new ones. For acquiring immunity, the best solutions are kept in a memory cell (see Fig. 10.8).

10.3.2 CHEMISTRY BASED METAHEURISTICS

10.3.2.1 Chemical Reaction Optimization (CRO)

Lam and Li [52] were proposed CRO which simulates the molecules interactions behavior in a chemical reaction in order to reach a low energy stable state. In CRO, the molecule represents a candidate solution and each molecule owns two energy types: Potential Energy (PE) and Kinetic Energy (KE). PE corresponds to a molecule objective function and KE denotes the tolerance of changing to a worst structure. Thus, in order to replace a molecule Θ with a new one $\grave{\Theta}$, the following condition should be satisfied:

$$PE_{\Theta} + KE_{\Theta} \geq PE_{\grave{\Theta}} \tag{10.6}$$

When examining some molecules in a closed container, it seems that the molecule collides with the container wall or with each other in a random movement. This lead to the CRO fundamental chemical reactions categories: on-wall ineffective collision (ON-WALL), decomposition (DEC), inter-molecular ineffective collision (INTER), and synthesis (SYN). For ON-WALL, the molecule hits the wall and rebounds resulting in losing some of KE and changing to a new one (after satisfies condition (10.6)). As the chemical reactions preserve the energy conservation law (energy cannot be created or destroyed, just transformed from one form to another), the lost energy is kept at the central energy buffer. DEC will take place if the molecule hit the wall and decomposes into pieces. This led to the emergence of two different cases: in the first case, the molecule has enough energy to complete DEC and yield energy to the new pieces. Otherwise, in the second case, DEC is not completed so that the molecule must get energy from the central energy buffer. For INTER, it is similar to ON-WALL but it results from the collision of two molecules then rebound away and there is no KE sent to the central energy buffer. The last type of reactions is SYN. Like INTER, SYN results from the collision of two molecules, but they consolidate together (see Fig. 10.9).

To prove the CRO capability of applying to NP-hard problems, CRO was tested on solving 23 instances of quadratic assignment problem (QAP). The results were compared with an oracle-based view

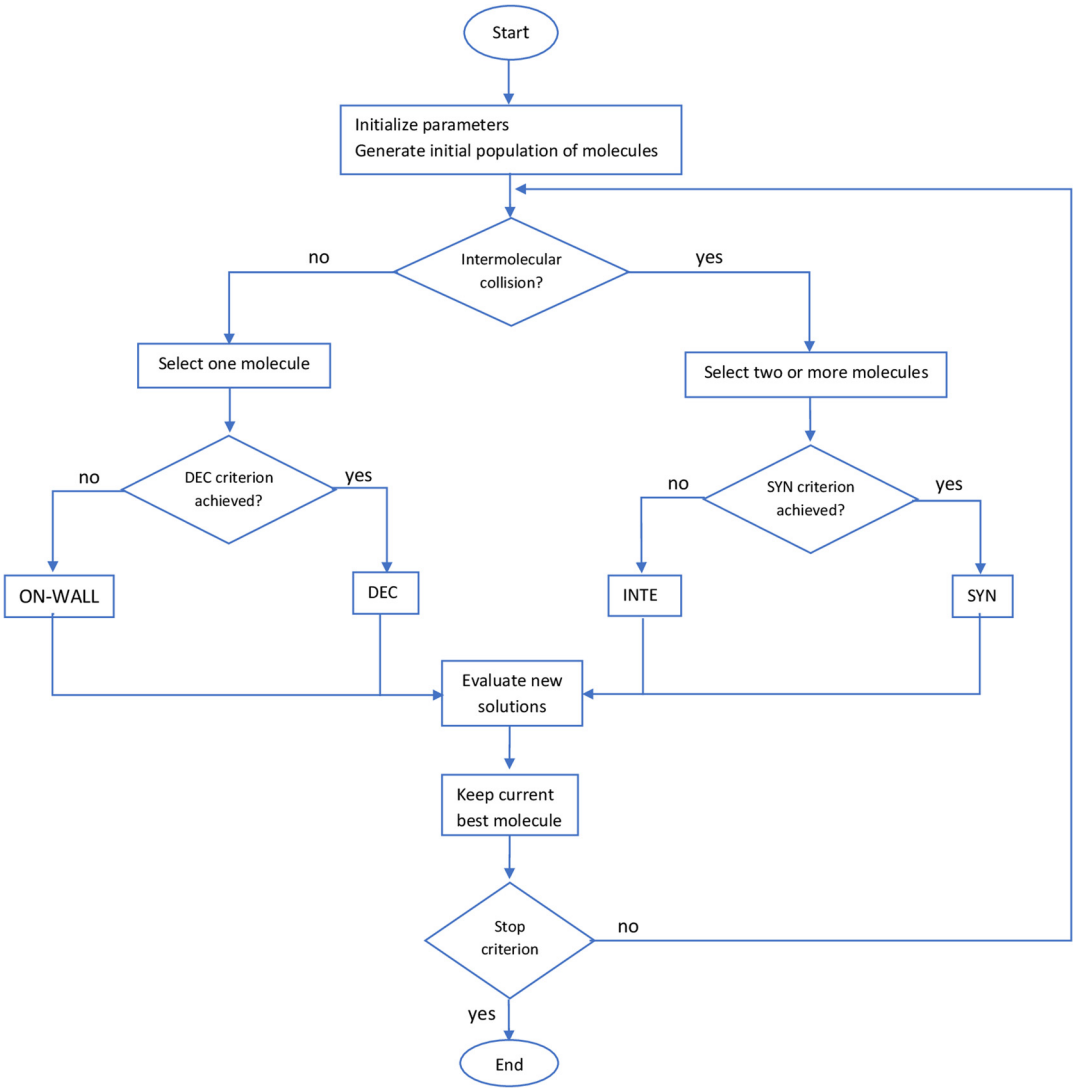

FIGURE 10.9

CRO pseudocode.

of computation of three popular metaheuristics that have solved QAP; Improved Simulated Annealing (ISA) [53], Fast Ant system (FANT) [54], and RobustTabu Search (TABU) [55]. In general, CRO was in a weak position in this comparison and need more improvements.

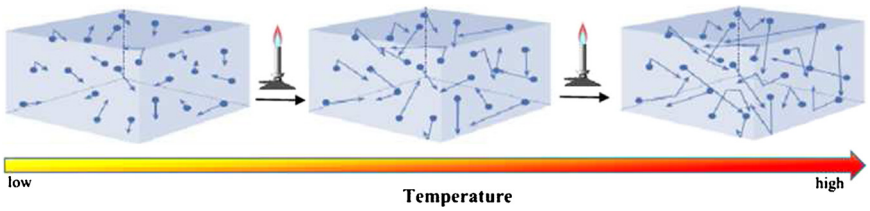

FIGURE 10.10

Impact of increasing temperature.

10.3.2.2 Gases Brownian Motion Optimization (GBMO)

The inspiration of GBMO [56] was taken from the motion of suspended fluid particles (turbulent rotational and Brownian motion). In GBMO, each candidate solution is presented by a molecule which has four specifications: position, mass, velocity, and turbulent radius. The molecules move towards the goal with respect to Brownian motion and it is evaluated according to its position. Besides, the environment temperature is a dominant parameter for controlling the kinetic energy and velocity of molecules and trade-off between diversification and intensification, i.e. the high temperature brisk kinetic energy and velocity of molecules and increase exploration while the low temperature decreases the random Brownian motion and increase exploitation (see Fig. 10.10). In contrast, the molecules turbulent rotational motion performs exploitation in high temperature and exploration in low temperature. The search process starts with generating an initial population of molecules randomly. Then, the turbulence radius is defined as a random number in a range [0, 1] and the environment temperature also defined (it is Better to start with high temperature and then reduced with the lapse of time). After that, the molecules positions and velocities are updated as:

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t) \tag{10.7}$$

$$v_i^d(t+1) = v_i^d(t) + \sqrt{\frac{3kT}{m}} \tag{10.8}$$

where $x_i^d(t)$ and $v_i^d(t)$ are the previous position and velocity respectively in the dth dimension of the search space ($1 \leq d \leq n$), k is Boltzmann's constant, T is the absolute temperature, and m is the molecule mass. Also, the molecule turbulent rotational motion which is a fluctuation in a specific radius is calculated by the circle chaos map. After that, the molecules are evaluated and according to this evaluation, their masses and the temperature are updated (see Fig. 10.11).

The proposed algorithm performance was tested in three ways. The first test, GBMO was tested on solving 18 instances of SAT problems. The results were compared with ICA, PSO, and GA and show that GBMO is efficient in solving SAT problems, especially for large instances. In addition, GBMO has a good performance for solving continues SAT problems and performed as GA discrete SAT versions (GASAT [57] and SATWAGA [58]). The second was tested on 7 well-known benchmark functions and in general, GBMO outperforms the competitors. However, the GBMO running time is little more than PSO. For the third test, two real world problems Lennard–Jones potential and Tersoff Potential Function Minimization problems were solved. GBMO proved its effectiveness and robust in solving

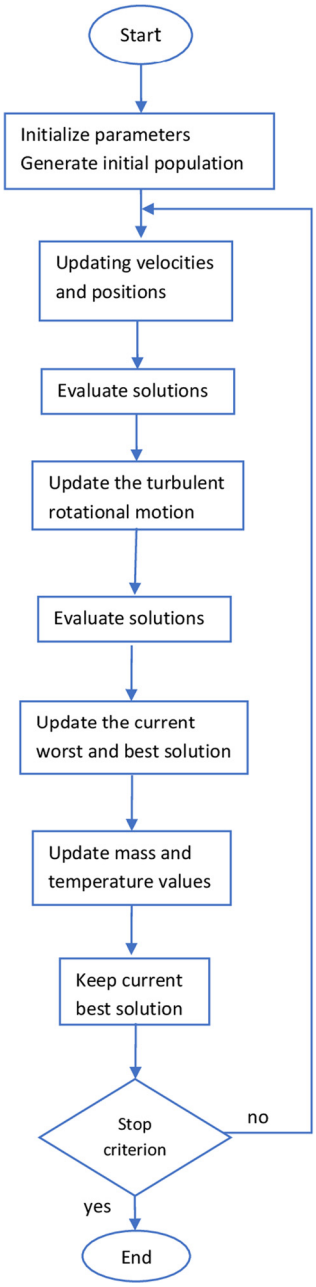

FIGURE 10.11

GBMO flowchart.

global optimization problems. Finally, the authors proposed a time complexity analysis of GBMO and the analysis show that the complexity of GBMO is $O(n)$.

10.3.3 MUSIC BASED METAHEURISTICS

10.3.3.1 Harmony Search (HS)

Geem et al. [59] proposed HS which imitates music improvisation process. In musical improvisation, each instrumentalist plays any tone within the possible range which together making one harmony vector. If all the tone makes a splendid harmony, this experience will be stored in each player's memory, and the possibility to play a good harmony is increased next time. In HS, each candidate solution is a vector. If all the values of a solution vector have a good harmony (high-fitness), this experience is preserved in each variable memory, and the possibility of making a good solution also increased next time.

The HS procedure [60] starts with the initialization of the harmony memory size (HMS) or the number of multiple vectors stored in the harmony memory, harmony memory considering rate (HMCR), and pitch adjusting rate (PAR). Then, the harmony memory (HM) is prepared as:

$$\left[\begin{array}{cccc|c} x_1^1 & x_2^1 & \cdots & x_n^1 & f(x^1) \\ x_1^2 & x_2^2 & \cdots & x_n^2 & f(x^2) \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ x_1^{HMS} & x_2^{HMS} & \cdots & x_n^{HMS} & f(x^{HMS}) \end{array}\right] \tag{10.9}$$

where x denotes a decision variable, n is the number of decision variables, and $f(x)$ is an objective function. HM solutions are evaluated and the corresponding values are sorted in the matrix in an ascending order. A new harmony vector is improvised according to HMCR, PAR, and random selection (1-HMCR), i.e. a new decision variable can be selected from HM with a probability HMCR or it can be randomly selected from the set of all candidate discrete values with a probability (1-HMCR). In addition, it can be slightly mutated by moving to neighboring values with probability PAR, like:

$$x_i^{New} = \begin{cases} x_i(k) \in \{x_i^1, x_i^2, \ldots, x_i^{HMS}\} & \text{if } r_1 < HMCR \\ x_i(k) \pm r_2 \times BW & r_3 < PAR \\ x_i(k) \in \{x_i(1), x_i(2), \ldots, x_i(k_i)\} & \text{otherwise} \end{cases} \tag{10.10}$$

where r_1, r_2, and r_3 are three different random numbers between [0, 1] and BW is the distance bandwidth. Finally, the newly generated vector is compared with the worst one. If the new vector is better, it will replace the worst vector. For a detailed analysis of HS performance, Milad [61] submitted a valuable study of HS. Also, many surveys were published as [62–65] (see Fig. 10.12).

10.3.3.2 Method of Musical Composition (MMC)

MMC [66,67] is a simulation of a musical society where the composers exchange information between themselves and their community to create a musical composition. Like HS, the main idea is creating a good musical artwork from the participation of composer's tunes. But, MMC is not a variant of HS because MMC follows a social system including exchange and learning methodology. In MMC

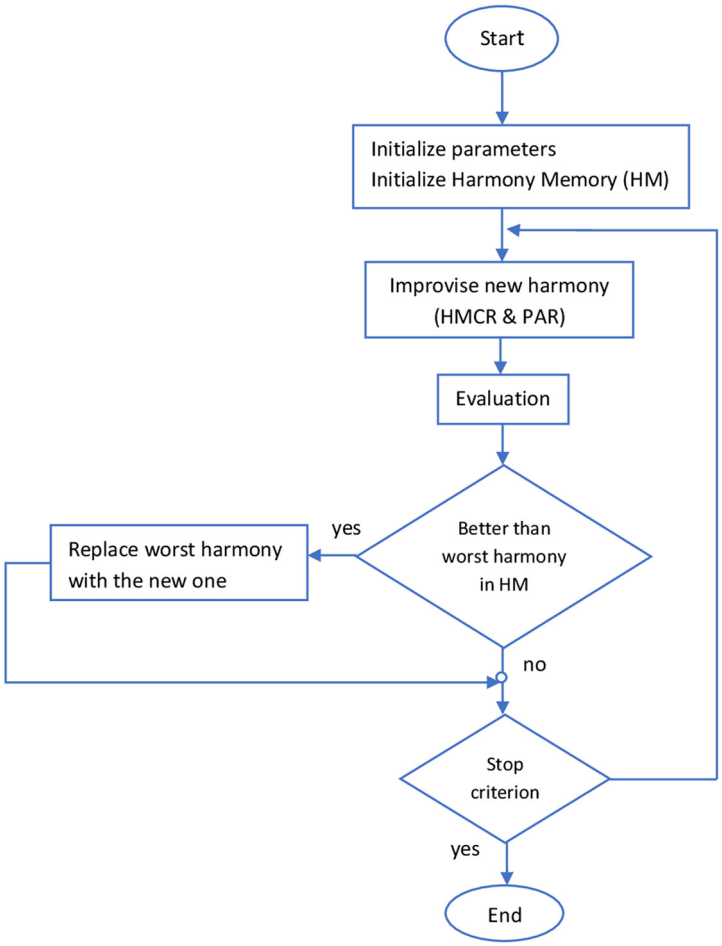

FIGURE 10.12

HS flowchart.

algorithm, each candidate solution is symbolized by a composer tune that is an n-dimensional vector of decision variables. In the initialization phase, each composer produces a set of random tunes which stored in a score matrix (like memory matrix in HS). Then, the information exchange is performed with respect to the interaction rules. In other words, the exchange is done if and only if there is a link between exchangers and the worst tune of the first composer is better than the worst tune of the second one. This phase divided into two subphases: links updating and information exchange. In the former subphase, the artificial society network is changed with respect to the previous. In the later subphase, each composer is provided with information about their environment to build his knowledge matrix (KM) which consists of the score matrix and the information from the environment. After that, a new tune is generated based on KM and for each composer, the score matrix is updated (see Fig. 10.13).

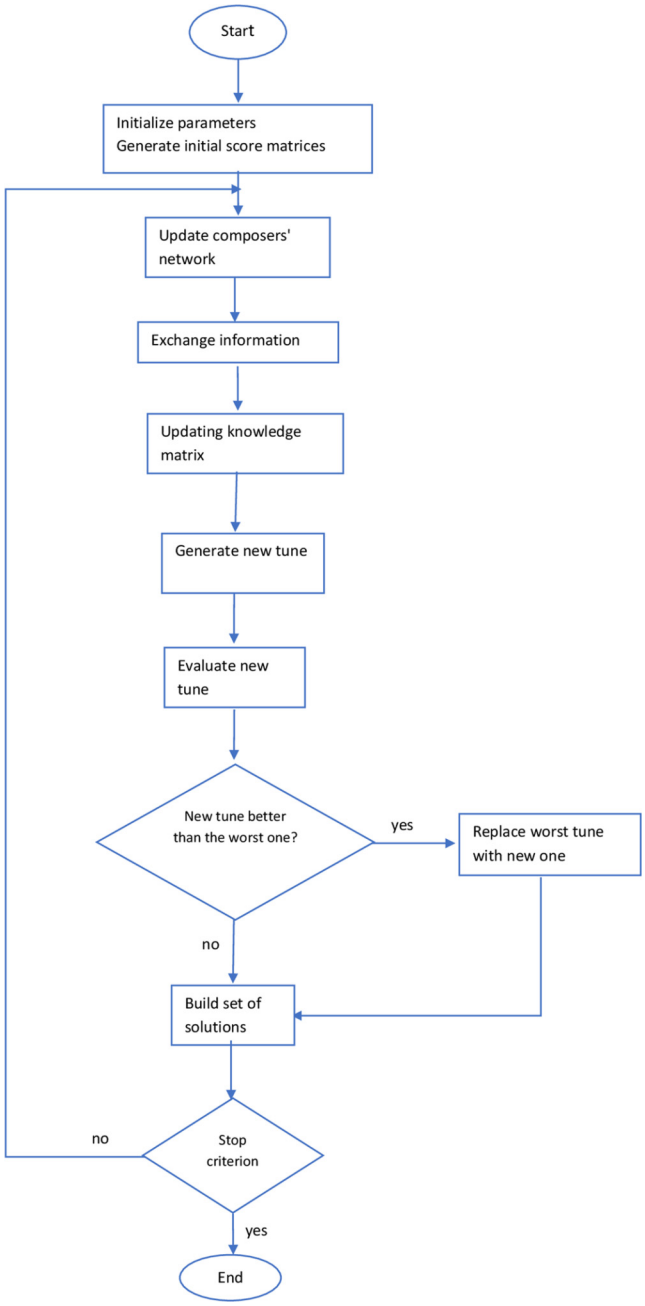

FIGURE 10.13

MMC flowchart.

In order to verify the efficiency of MMC, the authors tested the proposed algorithm on 13 continuous benchmark problems and the results compared with HS, improved HS [68], global-best HS [69], and self-adaptive HS [70]. The experimental results showed that MMC outperform the other methods, especially for the multi-modal functions.

10.3.4 MATH BASED METAHEURISTICS

10.3.4.1 Base Optimization Algorithm (BOA)

Salem [71] proposed BOA which depends on a combination of the basic arithmetic operators $(+, -, \times, \div)$. In the BOA first step, an initial population of solutions is randomly generated and the displacement parameter is defined. Then, each solution is evaluated and four possible solutions are calculated using the following formula:

$$x_i^{+}(j) = x_i(j) + \triangle \tag{10.11}$$

$$x_i^{-}(j) = x_i(j) - \triangle \tag{10.12}$$

$$x_i^{\times}(j) = \times_i(j) \times \triangle \tag{10.13}$$

$$x_i^{\div}(j) = x_i(j) \div \triangle \tag{10.14}$$

where $x_i(j)$ is the previous solution, $\triangle$ is the displacement parameter and all the calculated solutions values within a predefined range. Then, the four possible solutions are evaluated and the best one is selected to be the new candidate solution (see Fig. 10.14).

The author tested the proposed algorithm for solving using 8 benchmark functions including 2 unimodal and 6 multi-modal functions. The BOA results were compared with GA and ANTS [72] and outperformed the results of competitors in terms of success rate and solution quality.

10.3.4.2 Sine Cosine Algorithm (SCA)

Mirjalili [73] introduced SCA which derived its name from the use of sin and cos equations. Like population-based metaheuristics, the algorithm starts with the generation of initial population randomly and keeping the current best solution. During the search process, the following equation is used for updating the candidate solutions locations with respect to the current best solution as:

$$x_i^{(t+1)} = \begin{cases} x_i^t + (\delta_1 \times sin(\delta_2) \times |\delta_3 b_i^t - x_i^t|) & \text{if } \delta_4 < 0.5 \\ x_i^t + (\delta_1 \times cos(\delta_2) \times |\delta_3 b_i^t - x_i^t|) & \text{if } \delta_4 \geq 0.5 \end{cases} \tag{10.15}$$

where x_i^t and b_i^t are the previous position of a solution and the position of the current best solution respectively in ith dimension at tth iteration, $\delta_{(1...4)}$ are different random numbers in $[0, 1]$. Also, at each iteration the ranges of sin and cos functions are updated for more search space exploitation as following:

$$\delta_1 = c + t\frac{c}{T} \tag{10.16}$$

where c is a constant and T is the maximum number of iterations.

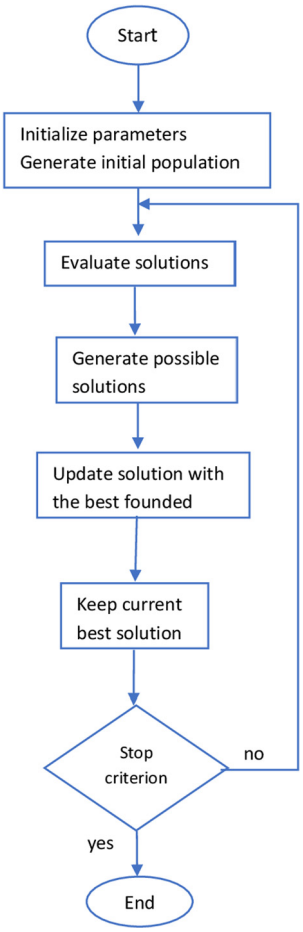

FIGURE 10.14

BOA flowchart.

It is notable that the random numbers $\delta_{(1...4)}$ are the dominance in SCA, i.e. δ_1 determines the movement direction, δ_2 dictates the length of the movement, δ_3 is a random weight for defining the destination randomly, and δ_4 switches between sin and cos equations. Also, changing the range of sin and cos functions is reflected on the updating of a solution position. In other words, the balancing between exploration and exploitation can be obtained by changing the range of sin and cos functions (see Fig. 10.15).

SCA was tested on popular benchmarks including unimodal, multi-modal, and composite functions and in addition, it was tested on the Wilcoxon ranksum test. The overall performance of SCA was compared with several algorithms as PSO, GA, BA, GSA, Flower Pollination Algorithm (FPA)

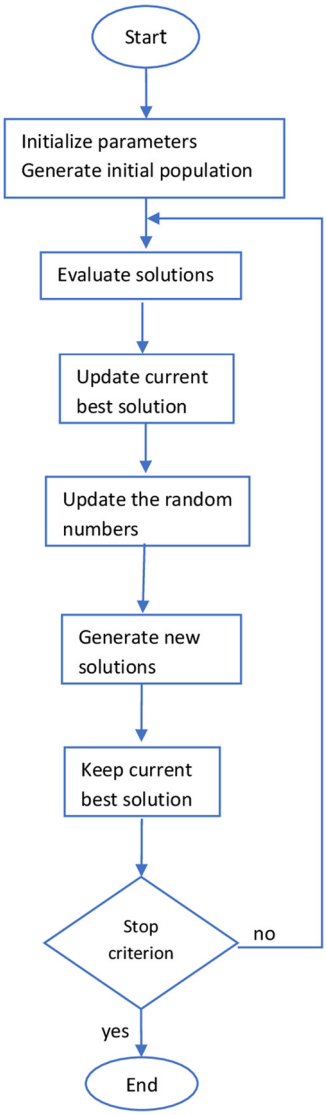

FIGURE 10.15

SCA flowchart.

[74], and Firefly Algorithm (FA) [75]. Moreover, the fewer parameters and the ability to extend to high dimensions make SCA a good choice to solve complex optimization problems such as airfoil design.

```
Generate initial solution x^c;
// R_max is the number of iterations
For r = 1 to R_max do
  While (termination criterion is not satisfied) do
      Compute x^n ∈ N(x^c);
      // creates a new solution x^n by adding a small perturbation to x^c
      Compute Δ;
      if (Δ < 0) then
          x^c = x^n ;
      else
          x^c = x^n with the probability P ;
  end while
  Reduce T;
end for
```

FIGURE 10.16

SA pseudocode.

10.3.5 PHYSICS BASED METAHEURISTICS

10.3.5.1 Simulated Annealing (SA)

SA was proposed by Metropolis et al. [76] to simulate the material cooling in a heat bath. After that, Kirkpatrick et al. [77] applied the idea to optimization problems, regardless the word "metaheuristic" was not known in those years. SA derived its name from the physical annealing process. Thus, like annealing initial state the algorithm is lenient and could move to a worse solution. At each iteration, the algorithm becomes more stringent for getting a better solution at each step, i.e. SA starts with a random initial solution x^c and high-temperature T. Then, another solution x^n is created randomly near the initial solution and the difference in the function values $\triangle$ is calculated as:

$$\triangle = f(x^n) - f(x^c) \tag{10.17}$$

If $\triangle$ is smaller, x^n automatically becomes the current solution from which the search will continue. Otherwise, the point is accepted with a probability:

$$P(\triangle, T) = e^{(-\triangle/T)} \tag{10.18}$$

Finally, T is reduced (see Fig. 10.16).

For proving the efficiency of SA, many studies of SA performance were submitted as [78–81].

10.3.5.2 Gravitational Search Algorithm (GSA)

GSA [47] was inspired from Newton's laws of gravity and motion. In GSA, each candidate solution is considered an object which has a position, inertial mass, active gravitational mass, and passive gravitational mass. The positions of objects represent solutions and their fitness is measured by their masses. Like PSO, the solution update depends on the velocity as Eq. (10.2). The higher performance objects with a greater gravitational mass have a large radius of effective attraction and hence a great intensity of attraction. So, the objects tend to move toward the best solutions (see Fig. 10.17). For more discussion, Nezamabadi-Pour [82] introduced a good survey on GSA.

```
Generate initial population
  While (stopping criterion not satisfied) do
      Evaluate Fitness of All Agents;
      Compute Gravitational Constant;
      Update Gravitational and Inertial Masses;
      Compute Total Force;
      Compute Acceleration and Velocity;
      Update Positions of Agents;
  End
  Return Best Found Solution;
```

FIGURE 10.17

GSA pseudocode.

10.3.6 SOCIAL AND SPORT BASED METAHEURISTICS

10.3.6.1 Teaching–Learning-Based Optimization (TLBO)

TLBO [83] simulates the influence of the teachers on the students in a classical school learning process, i.e. the teacher (the best solution) shares his knowledge with the students (solutions population) and his quality of teaching effect on the student's grades (fitness values). The TLBO learning process is divided into two main stages: Teacher phase and Learner phase (see Fig. 10.18). In the first phase, the best solution is selected to be a teacher and the mean of the students positions is calculated and shifted towards the teacher's position. Then, the new student position is calculated as:

$$x_i^{(t+1)} = x_i^t + r(x^* - TF.\bar{x}) \tag{10.19}$$

where x_i^t is the previous ith student position, r is a random number between $[0, 1]$, x^* is the current teacher position, $\bar{x}$ is the mean of the current population, and TF is the teaching factor that calculated randomly. In the second phase, the student i increase his knowledge by interacting with another student j that selected randomly. The two students i and j are compared according to three rules:

(1) If both solutions are feasible, then the solution with better fitness value is selected.
(2) If one solution is feasible and the other infeasible, then the feasible solution is selected.
(3) If both solutions are infeasible, then the solution having the minimum transgression of the feasibility constraints is selected.

The knowledge of the student i is modified if the knowledge of the selected one j was better. The modification of the student's knowledge is applied as:

$$x_i^{t+1} = \begin{cases} x_i^t + r(x_i^t - x_j^t) & \text{if } f(x_i^t) < f(x_j^t) \\ x_i^t + r(x_j^t - x_i^t) & \text{otherwise} \end{cases} \tag{10.20}$$

The authors showed that TLBO is an algorithm-specific parameter-less algorithm as it requires no special parameters (only require regular controlling parameters such as population size and a number of generations). To prove the TLBO efficiency, the authors compared it with Multimembered Evolutionary Strategy (M-ES) [84], Particle Evolutionary Swarm Optimization (PESO) [85], Cultural Differential Evolution (CDE) [86], Co-evolutionary Differential Evolution (CoDE) [87], and Artificial Bee Colony

```
Generate initial students ;
Evaluate students ;
Initialize the teacher;
Calculate the mean of all students;
While (termination criterion is not satisfied) do
// Teacher Phase
   For each student
     Update student position according to teacher position;
   end
   Evaluate new students ;
   Accept the new solutions better than the old ones;
// Learner Phase
  For each student
    Select another student randomly;
     If the selected student is better
        Update student position ;
      end
   end
   Evaluate new students ;
   Accept the new solutions better than the old ones;
   Update the teacher and the mean;
end
```

FIGURE 10.18

TLBO pseudocode.

(ABC) [88] for solving many constrained benchmark functions and mechanical design problems. In general, the TLBO results outperformed the competitors.

10.3.6.2 League Championship Algorithm (LCA)

LCA [89] emulates a sports championship where several teams play in a league for a number of seasons, i.e. like a real sports championship, a number of teams (solutions) participate in a league (population of solutions) and compete in pairs. Then, the match is analyzed by strengths/weaknesses/opportunities/threats (SWOT) analysis. The winner team is determined according to the good formulation of the team players and the strength of playing (fitness value). Whereupon, each team will enhance its performance by changing its game style by reformulating its players. The tournaments resume for a number of weeks until the ending of the season. For match analysis, there are 6 requisite assumptions:

(1) According to "playing strength" the winner team is determined.
(2) The one tournament result does not prognosticate the playing strength of a given team definitely.
(3) Each team can compete any other team with the same probability.
(4) The tie result is neglected, only win or loss result is considered.

```
Initialize parameters;
Generate league schedule;
Generate initial population; //team formulations
Evaluate the fitness of each team;
Determine winner/loser;
  While (stopping criterion not satisfied) do
      Generate new solution;
      Evaluate new solution;
      Update the current best solution ;
      If mod(number of iterations, league size − 1) == 0
        Generate new league schedule;
      End
  End
  Return best found solution;
```

FIGURE 10.19

LCA pseudocode.

(5) When two teams compete, any strength supports one to win has a corresponding weakness caused the loss of the other.

(6) A team just focuses on the next match without regard of the other subsequent matches. The reformation is done just based on the previous week performance.

The proposed algorithm starts with the initialization of initial population. Then, a league is scheduled, i.e. a single round-robin schedule is employed where each team competes another participant once in each season. The determination of winner/loser teams is performed randomly. For the forthcoming match, a new formulation of teams is performed considering the current best formation and the previous week SWOT analysis (see Fig. 10.19). Besides, the authors tried to include the tie case when new formations were done and adapting players transfer mechanism at the end of the season in order to increase the LCA convergence speed.

In order to test the validity of LCA, it was tested on 36 benchmarks and compared with GA, DE, PSO, and ABC algorithms. In general, the results showed that LCA was more effective than the competitors. In addition, LCA was compared with PSO and Dynamic multi-swarm particle swarm optimizer (DMS-PSO) [90] in solving 14 benchmark functions selected from the 2005 IEEE Congress on Evolutionary Computation (CEC2005) and the overall LCA performance was accepted. For more details about LCA related publications, see [91].

10.4 NON-METAPHOR BASED METAHEURISTICS

10.4.1 TABU SEARCH (TS)

Tabu Search (TS) was presented by Glover and McMillan [92], who first used the term "metaheuristic". Also, TS can be classified as EA because of its iterative searching. The main idea of TS prohibition is "tabu or taboo" of already visited search area from being visited again to promote diversification, i.e. in order to explore the solution space and avoid getting trapped in local optima, TS applies any local search procedure intensively. There are two main features included in TS: adaptive memory and responsive exploration. The former (called the tabu list) stores the history of the past performed actions

```
Create initial solution s;
Initialize tabu list τ;
While (stopping criterion not satisfied) do
    Determine complete neighborhood η of current solutions;
    Create best non-tabu solution ś from η;
    Switch over to solution ś;    // current solution s is replaced by ś
    Update tabu list τ;
    Update best found solution;    // if necessary update
end
return best found solution;
```

FIGURE 10.20

TS pseudocode.

at the time of search process to avoid the siege within cycles. The latter uses the former one to make the search process focuses on the good regions and best solution for more intensification and exploring the promising new regions for more exploration (see Fig. 10.20). For more discussion about TS, see [93–99].

10.4.2 VARIABLE NEIGHBORHOOD SEARCH (VNS)

VNS [100] is a stochastic metaheuristic that can be considered more advanced multi-neighborhood search compared to Variable Neighborhood Descent (VND) and Reduced Variable Neighborhood Search (RVNS) which are extensions of hill climbing algorithm [101]. The foundations of VNS is built upon the following facts:

(1) A local optimal for one neighborhood is not necessarily so for another.
(2) A global optimal is a local optimum regarding all possible neighborhoods.
(3) For many problems, all local optimal are relatively close to each other.

The last fact is obtained by empirical observations. However, it does imply that the local optimal solutions may have some useful information about the global optimal.

The main idea is to systematically or randomly explore several neighborhoods during the search for an optimal (or near-optimal) solution. VNS based on applying alternately two improvement procedures. The first is "shaking" which is done to get out of the corresponding valley. Also, a simple local search is applied to get from these neighbors the local optimal, or some more advanced procedures that explore several neighborhood structures. The second step is "neighborhood changing" in which VNS explores randomly outlying neighborhoods of the current solution and moves from there to a new one if and only if an improvement is made. In this way, keeps good variables and obtains promising neighbors (see Fig. 10.21). For more details about VNS, different studies have been proposed as [102–104].

10.4.3 PARTIAL OPTIMIZATION METAHEURISTIC UNDER SPECIAL INTENSIFICATION CONDITIONS (POPMUSIC)

POPMUSIC was introduced by Taillard and Voss [105] as metaheuristics for the exploitation of the large search space. In other words, POPMUSIC is a local search that can optimize large-scale optimization problems (especially combinatorial optimization problems) by dividing the problem into

```
Initial the set of neighborhood structures N_s , s = 1, ..., s_max ;
Generate initial solution x;
  While (stopping criterion not satisfied) do
      s = 1;
      While (s ≤ s_max ) do
        Select x' randomly from N_s ;   \\ Shaking
        Apply a local search based on x' to get local optimum x_l* ; \\ Local Search
        If x_l* is better than x
          x = x_l*;    \\ Neighborhood Changing
          s = 1;
        Else
          s = s + 1;
        End
      End
```

FIGURE 10.21

VNS pseudocode.

subproblems, i.e. the fundamental approach of POPMUSIC is the breaking down of the problem S into n sub-parts $(s_1, s_2, \ldots, s_n)$ and then formulate subproblem R which consists of a chosen part (seed part) and its $r < n$ neighbor parts according to a predefined distance. After the formulation of the subproblem, it will be solved using a metaheuristic or an exact algorithm. These procedures are repeated until improving all the subproblems (see Fig. 10.22).

POPMUSIC serves as a general frame comprises other search procedures such as Large Neighborhood Search [106], Local Optimizations (LOPT) [107], Adaptive Randomized Decomposition [108], etc. Therefore, the authors listed four general aspects that should be determined when applying POPMUSIC framework:

(1) The definition of the solution parts.
(2) The selection procedure of a part for avoiding the improvement repetition of a subproblem.
(3) The function that links between parts.
(4) The subproblem optimization algorithm.

To prove the efficiency of POPMUSIC, the authors applied POPMUSIC for solving two popular applications: the centroid clustering and balancing mechanical parts. The results showed that POPMUSIC is more efficient than VNS based methods when applied to the sum-of-squares clustering problem. For the problem of balancing mechanical parts, TS was embedded with POPMUSIC. The statistical analysis exposed that the POPMUSIC-based TS was more efficient than the original TS for large instances of this problem.

10.5 VARIANTS OF METAHEURISTICS

10.5.1 UPGRADING OF METAHEURISTICS

Generally, metaheuristic is a mathematical process that iteratively new solution x^{t+1} from the previous on x^t. The algorithm consists of a vector of k parameters and m random numbers that operate the stochastic search. The solution can be formulated as:

$$x_n^{t+1} = A(x_n^t, P_k, R_m) \tag{10.21}$$

```
Generate an initial solution S randomly;
Divide S into n parts where N = {s1, s2, ... , sn};
Set O = ϕ;
While (O ≠ N) do
    Select si ∉ O;
    Create a subproblem R consist of r < n neighbour parts ;
    Improve R using selected optimizer;
    If R improved
      Update S ;
      O ← ∅ ;
    Else
      O ← {si} ∪ O;
    End
Return S ;
```

FIGURE 10.22

POPMUSIC pseudocode.

where A is a nonlinear mapping from the current solution x_n^t to the new solution vector x_n^{t+1}, $P_k = [p_1, p_2, \ldots, p_k]$ is the vector of algorithm parameters, and $R_m = [r_1, r_2, \ldots, r_m]$ is the vector of random numbers.

Although the basic metaheuristics could be very efficient, further transformations and improvement are submitted from time to time so as to increase the convergence of the algorithm and the solution quality. There are different modifications and improvements can be done to metaheuristic in terms of parameters or randomization operators. Next, some examples will be discussed.

10.5.1.1 Adaptive Metaheuristics

The random step size or the search range can be automatically adjusted with respect to algorithm progress to prevent premature convergence. Examples: MAX–MIN ant system [109], Adaptive particle swarm optimization algorithm [110], Adaptive simulated annealing [111], Adaptive Firefly Algorithm [112], Improved Harmony Search [68], Modified Cuckoo Search [113], Whale Optimization Algorithm [114], Improved Grey Wolf Algorithm [115], etc.

10.5.1.2 Chaotic Metaheuristics

Chaos maps (Logistic map, Chebyshev map, Tent map, etc.) are evolution functions that generate a deterministic bounded sequence of random numbers based on the initial condition with different time domain (continuous or discrete). Besides, chaos maps can replace the metaheuristic's random sequence generator as they are random-like, non-period, and non-converging for parameter adaptation. Recently, chaos-based metaheuristics are widespread as they utilize more randomness and high convergence rate, e.g. Chaotic particle swarm optimization [116], Chaotic harmony search [117], Chaotic Genetic Algorithm [118], Chaotic League Championship Algorithm [119], etc.

10.5.1.3 Gaussian Based Metaheuristics

The Gaussian distribution is used as a fine tuning procedure. In addition, some algorithms replace regular solution updating with sampling from a Gaussian distribution. For example, Bare-bones particle swarms [120], Gaussian firefly algorithm [121], Gaussian Bare-bones differential evolution [122], Bare-bones teaching-learning-based optimization [123], etc.

10.5.1.4 Metaheuristics Acceleration

This modification aims to speed up the algorithm by omitting unimportant parameters or adding new parameters. The most popular one is accelerated PSO [124]. In accelerated PSO, the individual best solution is eliminated and the new solution simply depends on the global best solution. Other examples are Accelerated Artificial Bee Colony [125], Accelerated mine blast algorithm [126], Accelerated Biogeography-based optimization [127], etc.

10.5.2 METAHEURISTICS ACCLIMATIZATION

10.5.2.1 Multi-Objectives Metaheuristics

The conventional single-objective metaheuristics become inadequate for a wide range of optimization problems that have multiple conflicting objectives. So that, the need of multi-objective optimization has been raised. Multi-objective optimization can reunite between a set of alternatives instead of a single/best solution of single-objective metaheuristic "Pareto front" which subject to a set of constraints. Also, it can be also called "vector optimization", "multi-criteria optimization", "multi-attribute optimization" or "Pareto optimization". For more information see [128–130]. Next section NSGA-II and MOPSO-CD will be discussed as examples.

Non-dominated Sorting Genetic Algorithm (NSGA-II)

NSGA-II is a modified version of NSGA [131] that first introduced by Deb et al. [132] for solving several constrained problems such as a five-objective seven-constraint nonlinear problem. The difference between these two versions is that NSGA-II is faster, has better sorting algorithm, includes elitism which preserves an already founded Pareto optimal from deleting, uses explicit diversity preserving mechanism, and its complexity is at most $O(MN^2)$, while the NSGA complexity is $O(MN^3)$ (where M is the number of objectives and N is the population size). NSGA-II incorporates standard GA (select, crossover, and mutation) with non-dominated sorting and new fitness value "Crowding Distance" which is assigned in order to measure the density of solutions surrounding a particular solution.

Multi-Objective Particle Swarm Optimization With Crowding Distance (MOPSO-CD)

First MOPSO algorithm was proposed by Moore and Chapman [133]. There are two different approaches for designing MOPSO algorithms [134]: The first one is the separated consideration of each objective function, i.e. in this approach, each particle is evaluated only for one objective function at a time, and the selection of best positions is performed similarly to single-objective PSO. In this case, the main difficulty is properly manipulating information coming from each objective function to guide the particles towards Pareto optimal solutions. The second approach is the evaluation of each particle for all objective functions and based on the concept of Pareto optimality, they produce non-dominated best positions which are used to guide the particles. In this approach, the determination of best positions is not easy, as there can be many non-dominated solutions in the neighborhood of a particle, but only one is usually selected to contribute in the velocity update. To transact with the growing size of particles best positions (Pareto front), a new external repository is used, i.e. each particle experiences will be stored in an external repository after each iteration. Unfortunately, the external repository also has a bound size so, the replacement of existing solutions with new ones should be considered. MOPSO-CD first introduced by Raquel and Naval [135]. The crowding distance mechanism has been added with a

mutation operator to preserve the diversity of non-dominated solutions in the external repository. The overall complexity of MOPSO-CD is $O(MN^2)$.

10.5.2.2 Metaheuristics Discretization

A variety of optimization problems cannot be handled by continuous metaheuristic. Thus, many researchers developed several discretization methods to accommodate binary and integer-valued combinatorial problems such as Traveling Salesman Problem, Facility Location Problem, Makespan Scheduling Problem, and Knapsack problem, etc. The main discretization methods are:

(1) Sigmoid Function (SF) is a popular discretization method that transforms continuous search area into a binary one by using the following equation:

$$x_{ij} = f(x) = \begin{cases} 1 & \text{if } rand() \leq \frac{1}{exp(-x_{ij})} \\ 0 & \text{otherwise} \end{cases} \tag{10.22}$$

Examples: Binary particle swarm optimization (BPSO) [136], Discrete Firefly [137], Binary Cat Swarm Optimization (BCSO) algorithm [138], Binary black holes algorithm [139], Binary flower pollination algorithm (BFPA) [140], etc.

(2) Random-key (RK) uses a random numbers vector thereafter assigns a weight to each vector element (in ascending order). These weights are used to produce one permutation as a solution. i.e. $\vec{x}_i = [0.2, 0.03, 0.8, 0.47, 0.19]$ can be transformed to $\vec{x}_i = [3, 1, 5, 4, 2]$. Examples: Genetic Algorithm With Random-Key [141], Improved Shuffled Frog-Leaping Algorithm [142], Random-Keys Golden Ball Algorithm [143], Random-Key Cuckoo Search [144], etc.

(3) Smallest Position Value (SPV) is mostly used in task scheduling problems. First, the position vector is transformed to a permutation $S_k^i = [s_1^i, s_2^i, \ldots, s_n^i]$. Then, the sequence of operation $R_k^i = [r_1^i, r_2^i, \ldots, r_n^i]$ is calculated as:

$$R_k^i = S_k^i \; mod \; m \tag{10.23}$$

where n is the number of tasks and m is the number of resources. Examples: PSO [145], Shaking Optimization Algorithm (SOA) [146], Discrete Firefly Algorithm [147], Memetic Gravitation Search Algorithm (MGSA) [148], etc.

For a detailed discussion on the discretization methods of metaheuristics see [149].

10.5.2.3 Metaheuristics Continuousization

A wide range of engineering design and control problems are formulated as a continuous problem. In other words, they can be defined as unconstrained optimization problems, like:

$$\begin{aligned} &min/max \quad f(X) \quad , X = x_1, x_2, \ldots, x_n \\ &s.t. \quad X \in [A, B] \end{aligned} \tag{10.24}$$

where $f(X)$ is the objective function, X is the set of decision variables, $A = a_1, a_2, \ldots, a_n$ and $B = b_1, b_2, \ldots, b_n$ are the lower and the upper bounds of the decision variables.

Many metaheuristics have proved their efficiencies in the discrete domain. In order to get benefit from their advantages in a continuous domain, the metaheuristics should be modified to be able to accommodate floating point numbers. There are two approaches: the first approach is the usage of transform methods like discretization, and the second approach is to estimate the distributions (mixture) parameters. The most popular real-coded metaheuristic is the real-coded GA (RCGA) [150]. In RCGA, a candidate solution (chromosome) is a vector of floating point numbers instead of binary bits which may cause precision losing. Also, Lam et al. [151] proposed a real-coded CRO (RCCRO) which employs the Gaussian distribution to produce perturbations to search in the continuous domain.

10.5.3 HYBRIDIZATION OF METAHEURISTICS

As mentioned before, there are no metaheuristics can solve all problems. Hybridization of metaheuristics gets benefit from the point of power in each algorithm. It can be done between different techniques in low or high level according to the interference degree between both algorithms components, i.e. high-level hybridization indicates that low interference between the internal work of hybridized algorithms while low-level hybridization means that only a metaheuristic function is exported to another metaheuristic. During the search process, the hybridized algorithms may collaboratively exchange information or integrated with a master one that operates the search procedure. Furthermore, the algorithm execution sequence must be taken into account. To sum up, hybridization can be categorized based on different characteristics such as:

- With what hybridization is (e.g. metaheuristics, machine learning, exact methods, etc.).
- The order of metaheuristics execution (sequential, interleaved, or parallel).
- The control strategy (Collaborative, Integrative).
- The hybridization level (high, low).

Next, various hybridization examples will be discussed. For more information, see [152–155].

10.5.3.1 Sequential and Interleaved Hybridization

ANGEL Algorithm

ANGEL [156] combined ACO, GA, and a local search method (LS). In the presented hybridization, the initial population was generated by ACO for a good start. Then, the generated population was processed by GA. In addition, a feedback mechanism was considered between GA and ACO. The LS is applied when a feedback is provided to ACO by GA via entirely updating pheromone during the search process.

MKF-Cuckoo Algorithm

Binu et al. [157] combined the capabilities of cuckoo search algorithm (CS) [158] with the multiple kernel-based fuzzy c means algorithm (MKFCM) [159] to introduce MKF-Cuckoo. The main goal of the proposed algorithm is to find the best centroid in the search area. In MKF-Cuckoo, the candidate solutions are encoded by choosing random centroid taken from the input dataset that represented as a matrix. In addition, the fitness function employs fuzzy membership function to find the minimum

kernel-based distance between the data points with its nearest neighbor cluster centroid. Finally, the procedures of CS are applied.

FPA With Clonal Selection Algorithm (CSA)

In the proposed algorithm [160], exploitation phase of FPA (local pollination) was combined with the clonal selection methodology which mimics the theoretical immunology principles, i.e. if the switching condition directs the algorithm to local search, a population of the best antibodies (solutions) is selected from the current population. Then, the higher affinities antibodies are cloned in order to generate more antibodies against the antigen (objective function). Those antibodies are not cloned are replaced by new ones via local pollination. Also, a binary version of the proposed algorithm was presented in [161].

10.5.3.2 Metaheuristics in Parallel Schemas

The usual metaheuristic consecutive search structure became improper for handling sophisticated real-life optimization problems which may be dynamic or high-dimensional. For that, the recent trends are preferred to use parallelism [162] in solving such problems so as to take parallelism advantage high-quality solutions with low search time. For example: Parallel GA [163], Parallel artificial bee colony [164], Parallel GRASP [165], etc.

It is worth mentioning that the parallel implementation needs different components are needed as multi-core processors, networks, data storage, etc. In order to employ metaheuristics in parallel, there are two main frameworks: Trajectory-based and population-based models. Recently, several models of parallelism are hybridized in order to more efficiency. Next, the two basic models are discussed. For more information see [166,167].

Trajectory-Based Models

As discussed before, the behavior of trajectory-based metaheuristics is updating one solution by another from its neighbors. For applying this type of metaheuristics in the parallel environment, three basic search models are used: parallel moves, parallel multi-start, and move acceleration models. For the former model, the current solution of one metaheuristic (master) is duplicated and distributed to others at each iteration and the obtained results back to the master. While in the second model includes launching several trajectory-based metaheuristics for the same problem. In the last model, the fitness evaluation of each solution is performed in parallel because that the objective function is also parallelized.

Population-Based Models

Population-based metaheuristics indicate that the whole population is updated at each iteration. So that in many cases researchers employ a pool of processors to reduce time-consuming. In parallelization, there are three main models: master-slave, distributed, and cellular models. In the first model, the master metaheuristic plays the selection role while the others (slaves) are responsible for updating and evaluation of the solutions. The next models are popular than the former. For distributed model, the population of solutions is divided into subpopulations. A population-based metaheuristic is applied to each subpopulation. Also, the solutions are exchanged among these subpopulations for more diversification. In the last model, the solutions updated by the interaction with the near neighbors for more exploration.

FIGURE 10.23

Applications of weld beam (all photos are available in http://www.gunungsteel.com/).

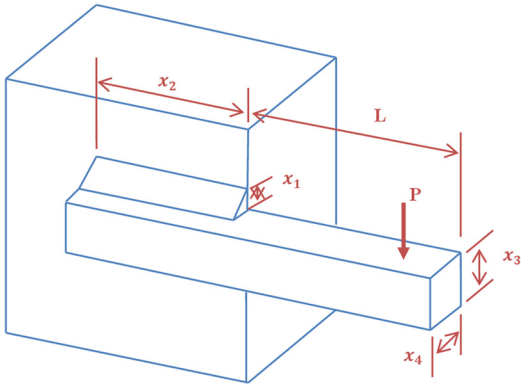

FIGURE 10.24

Weld beam design.

10.6 A CASE STUDY: WELD BEAM DESIGN PROBLEM

10.6.1 WELD BEAM DESIGN PROBLEM

Welded Beams are structural steel parts composed of a web and two flanges. They are usually used in engineering construction and heavy industries (see Fig. 10.23). Weld Beam Design problem [168] can be defined as: Find feasible dimensions of a Welded Beam x_1, x_2, x_3, and x_4 (see Fig. 10.24) that minimize the total manufacturing cost subject to a set of constraints on shear stress (τ), bending stress (σ), buckling load (P_C), end deflection (δ), side constraints. Mathematically, the problem can

be formulated as follows:

$$
\begin{aligned}
min \quad & f(X) = 1.10471x_1^2x_2 + 0.04811x_3x_4(L + x_2) \\
s.t. \quad & \\
& g_1(X) = \tau(X) - \tau_{max} \leq 0 \\
& g_2(X) = \sigma(X) - \sigma_{max} \leq 0 \\
& g_3(X) = x_1 - x_4 \leq 0 \\
& g_4(X) = 0.10471x_1^2 - 0.04811x_3x_4(L + x_2) - 5 \leq 0 \\
& g_5(X) = 0.125 - x_1 \leq 0 \\
& g_6(X) = \delta(X) - \delta_{max} \leq 0 \\
& g_7(X) = P - P_C(X) \leq 0 \\
& \text{where} \\
& \tau(X) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \\
& \tau' = \frac{P}{\sqrt{2}x_1x_2} \\
& \tau'' = \frac{MR}{J} \\
& M = P(L + \frac{x_2}{2}) \\
& R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2} \\
& J = 2(\sqrt{2}x_1x_2(\frac{x_2^2}{12} + (\frac{x_1 + x_3}{2})^2)) \\
& \sigma(X) = \frac{6PL}{x_3^2x_4} \\
& \delta(X) = \frac{4PL^3}{Ex_3^3x_4} \\
& P_c(X) = \frac{4.013\sqrt{E(x_3^2x_4^6/36)}}{L^2} \\
& P = 6000lb, L = 14in, E = 30 \times 10^6 psi, G = 12 \times 10^6 psi \\
& \tau_{max} = 13600psi, \sigma_{max} = 30000psi, \delta_{max} = 0.25in \\
& 0.1 \leq x_1 \leq 2, 0.1 \leq x_2 \leq 10, 0.1 \leq x_3 \leq 10, 0.1 \leq x_4 \leq 2
\end{aligned} \tag{10.25}
$$

Table 10.1 Experiment descriptive statistics

Algorithm	Minimum	Maximum	Mean	Std. Deviation	Average time (sec.)	Percentiles		
						25th	50th (Median)	75th
GA	1.78718	2.69881	2.0879626	0.2528459	19.7757	1.860510	2.075245	2.211833
PSO	1.72485	1.84167	1.7645672	0.0413752	3.57712	1.724852	1.748400	1.814293
SA	1.73543	1.81303	1.7713245	0.0189966	0.49096	1.757852	1.768612	1.782692
HS	1.97889	3.20092	2.6348878	0.2703827	7.3932	2.477490	2.723412	2.821445
TLBO	1.72485	1.72485	**1.7248523***	0.000	30.7576	1.724852	1.724852	1.724852
BBO	1.81484	2.58478	2.1721346	0.2463477	20.1665	1.979959	2.119732	2.414904
FPA	1.72493	1.72573	1.7252702	0.0001645	12.5984	1.725175	1.725257	1.725376
MVO	1.72595	1.74646	1.7312617	0.00476822	9.02517	1.727718	1.730086	1.732965
SCA	1.76807	1.85603	1.8155614	0.02413824	7.28464	1.797270	1.815022	1.833938
WOA	1.74513	2.12090	1.8589518	0.08298639	10.7514	1.808839	1.835831	1.887613

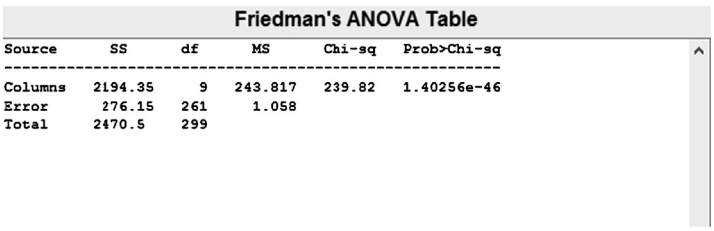

FIGURE 10.25

Friedman test results.

10.6.2 EXPERIMENTAL RESULTS

In this subsection, 10 metaheuristics are applied to the previously discussed problem include: GA (which is modified to real-coded version), PSO, SA, HS, TLBO, Biogeography-Based Optimization (BBO) [46], Flower pollination algorithm (FPA) [74], Multi-Verse Optimizer (MVO) [169], SCA, Whale Optimization Algorithm [114]. Table 10.1 shows the statistical results of the experiment. All experiments are carried out on a 64-bit operating system with a 2.60 GHz CPU and 6 GB RAM. For parameter settings, the number of search agents is set to 100 agents and the maximum iterations are set to 2000 iterations. (The other parameters of each algorithm is kept as default.) Besides, the experimental results are analyzed with nonparametric Friedman test [170] to show the differences in performance between the compared algorithms. As shown in Fig. 10.25, the p-value is less than the level of significant ($\alpha = 0.05$); thus, the compared algorithms are different in performance. Fig. 10.26 Friedman ranked mean and average time values of each metaheuristic. As shown, the performance of metaheuristics is varied in time and solution quality, e.g. the performance of TLBO is superior for the given problem as it is the only one that can reach the optimal solution. On the other hand, it has the worst average time. Also, SA is the fastest but it can not reach the optimal solution. Thus, a metaheuristic is chosen according to the need of solution quality and speed.

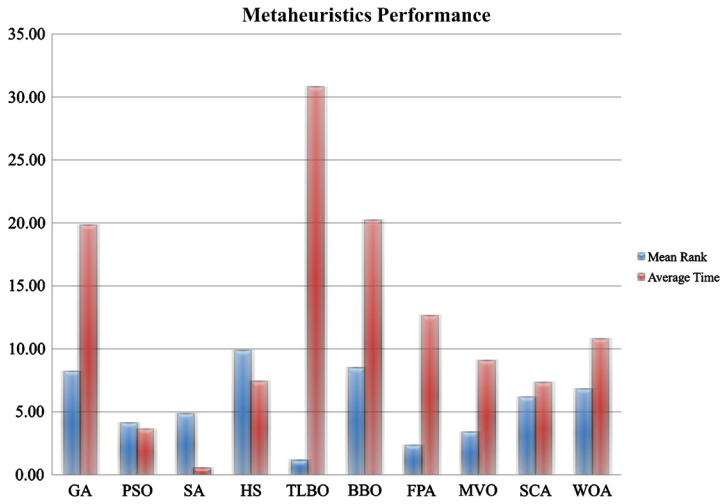

FIGURE 10.26

Metaheuristics performance (ranked mean and time).

10.7 LIMITATION AND NEW TRENDS

The optimization process is finding the best solution to a given problem. Thus, the main difficulty facing metaheuristics is how to deal with this problem. Although there is an enormous number of metaheuristics that have been proposed, only a few metaheuristics significantly achieved the desired success rate such as GA, HS, and CS, etc. Particularly, the population-based metaheuristics are widely used as they can adapt to large-scale optimization problems. As mentioned, metaheuristics are problem-dependent algorithms. Hence, the question is "what is the best definition of the algorithm parameters according to the type and size of the problem search space?". Besides, the selection of the appropriate metaheuristic algorithm is another challengeable task. To overcome these issues, recent trends tend to liberalize the metaheuristic techniques from the prototyping and parameter definition restriction, e.g. hyperheuristics [171,172] which denote the heuristic for heuristics selection. Also, de Melo and Banzhaf [173] introduced a new algorithm called Drone Squadron Optimization that generates the actual code on-the-fly by a cloud hyperheuristic. Papa et al. [174] proposed an open source library called LibOPT for implementing and prototyping of nature-inspired metaheuristics.

10.8 CONCLUSION

In this chapter, a bird's eye view of all metaheuristics related issues is taken. Various metaphor based and non-metaphor based metaheuristics are discussed. Taking into account that the exhibited taxonomies of metaheuristics are not mandatory and it is depending on the researcher perspectives. So, some algorithms can be categorized into different categories at once [5]. Often, non-metaphor based metaheuristics seem to be incomplete methodologies and almost their efficiency appears when it em-

bedded with other algorithms. Meanwhile, using too many metaphors as an inspiration may lead the area of metaheuristics away from scientific rigor [175].

Moreover, the chapter reviews some variants of metaheuristics. In order to provide new researchers with an overview of metaheuristics and guide those to find their starting point. Besides, a real-time case study "Welded Beam Design Problem" is solved with 10 different metaheuristics and the experimental results are statistically analyzed with non-parametric Friedman test. The statistical result indicates that performance of metaheuristics is different in solution quality and time. Metaheuristics perform differently in various types of problems. One may perform better than the other in a particular problem and worse in other sets of problems. For the given problem, TLBO is superior in obtaining the optimal solution.

Besides previously discussed metaheuristics, the investigations on metaheuristics are still being done and new metaheuristics are being developed continually. For future works, we suggest developing a unified platform for analyzing, evaluating, and comparing metaheuristics.

Appendix Metaphor Based Metaheuristics

Biology-Based Algorithms (Evolutionary, Swarm Intelligence, and Artificial Immune Systems)	
Algorithm	**Reference**
Evolutionary Programming	Fogel (1966)
Evolutionary Strategies	Rechenberg (1973)
Genetic Algorithm	Holland (1975)
Memetic Algorithm	Moscato (1989)
Genetic Programming	Koza (1990)
Ant Colony Optimization	Dorigo (1992)
Particle swarm optimization	Eberhart & Kennedy (1995)
Differential Evolution	Storn & Price (1997)
Clonal Selection Algorithm	De Castro & Von Zuben (2000)
Gene Expression	Ferreira (2001)
Marriage In Honey Bees	Abbass (2001)
Bacterial foraging	Passino (2002)
Artificial Immune Network	De Castro & Timmis (2002)
Queen-Bee Evolution	Jung (2003)
Shuffled Frog Leaping Algorithm	Eusuff & Lansey (2003)
B-Cell Algorithm	Kelsey & Timmis (2003)
Glowworm Swarm Optimization	Krishnanand & Ghose (2005)
Bee Colony Algorithm	Teodorović & Dell'Orco (2005)
Cat Swarm Optimization	Chu et al. (2006)
Invasive Weed Optimization	Mehrabian & Lucas (2006)
Seeker Optimization Algorithm	Dai et al. (2007)
Artificial bee colony	Karaboga & Basturk (2007)

Biology-Based Algorithms (Evolutionary, Swarm intelligence, and Artificial Immune Systems)	
Algorithm	**Reference**
Monkey Search	Mucherino & Seref (2007)
Good Lattice Swarm Algorithm	Su et al. (2007)
Fast Bacterial Swarming Algorithm	Chu et al. (2008)
Firefly Algorithm	Yang (2008)
Biogeography-Based Optimization	Simon (2008)
Fish-School Search	Bastos-Filho et al. (2008)
Roach Infestation Algorithm	Havens et al. (2008)
Paddy Field Algorithm	Premaratne et al. (2009)
Human-Inspired Algorithm	Zhang et al. (2009)
Group Search Optimizer	He et al. (2009)
Virus Optimization Algorithm	Cuevas et al. (2009)
Cuckoo Search Algorithm	Yang & Deb (2009)
Hunting Search algorithm	Oftadeh (2010)
Termite Colony Optimization	Hedayatzadeh et al. (2010)
Bat Algorithm	Yang (2010)
Hierarchical Swarm Model	Chen et al. (2010)
Eagle Strategy	Yang & Deb (2010)
Consultant-Guided Search	Iordache (2010)
Bioluminescent Swarm Optimization Algorithm	de Oliveira (2011)
Asexual Reproduction Optimization	Mansouri et al. (2011)
Cuckoo Optimization Algorithm	Rajabioun (2011)
Eco-Inspired Evolutionary Algorithm	Parpinelli & Lopes (2011)
Krill Herd Algorithm	Gandomi & Alavi (2012)
Migrating Birds Optimization	Dumana et al. (2012)
Weightless Swarm Algorithm	Ting et al. (2012)
Lion Pride optimizer	Wang et al. (2012)
Wolf Search Algorithm	Tang et al. (2012)
Blind, Naked Mole-Rats Algorithm	Shirzadi & Bagheri (2012)
Japanese Tree Frogs Calling	Hernández & Blum (2012)
Flower pollination algorithm	Yang (2012)
Differential Search Algorithm	Çivicioglu (2012)
Great Salmon Run	Mozaffari et al. (2012)
Unconscious Search Algorithm	Ardjmand & Amin-Naseri (2012)
Swallow swarm optimization Algorithm	Neshat et al. (2013)
Egyptian Vulture Optimization Algorithm	Sur et al. (2013)
Green Heron Optimization Algorithm	Sur & Shukla (2013)
Dolphin Echolocation	Kaveh & Farhoudi (2013)
Artificial Cooperative Search	Civicioglu (2013)
Atmosphere Clouds Model Optimization	Yan & Hao (2013)
Seven-Spot Ladybird Optimization	Wang et al. (2013)
Amoeboid Organism Algorithm	Zhang et al. (2013)

Biology-Based Algorithms (Evolutionary, Swarm intelligence, and Artificial Immune Systems)	
Algorithm	**Reference**
Coral Reefs Optimization Algorithm	Salcedo-Sanz et al. (2014)
Symbiotic Organisms Search	Cheng & Prayogo (2014)
Group Counseling Optimizer	Eita & Fahmy (2014)
Shark Smell Optimization Algorithm	Abedinia et al. (2014)
Grey wolf optimizer	Mirjalili et al. (2014)
Dispersive Flies Optimization	Al-Rifaie (2014)
Chicken Swarm Optimization Algorithm	Meng et al. (2014)
Social Spider Optimization	Cuevas & Cienfuegos (2014)
Elephant Herding Optimization	Wang et al. (2015)
Weighted Superposition Attraction	Baykasoğlu & Akpinar (2015)
Lion Optimization Algorithm	Yazdani & Jolai (2015)
Prey-predator algorithm	Tilahun & Ong (2015)
Moth-Flame Optimization Algorithm	Mirjalili (2015)
Artificial Algae Algorithm	Uymaz et al. (2015)
African Buffalo Optimization Algorithm	Odili et al. (2015)
Monarch Butterfly Optimization	Wang et al. (2015)
Tree-seed algorithm	Kiran (2015)
Butterfly Algorithm	Arora & Singh (2015)
Locust search	Cuevas et al. (2015)
Runner-Root Algorithm	Merrikh-Bayat (2015)
Ant Lion Optimizer	Mirjalili (2015)
Earthworm optimization algorithm	Wang et al. (2015)
Virus colony search	Li et al. (2016)
Cricket algorithm	Canayaz & Karci (2016)
Root Tree Optimization Algorithm	Labbi et al. (2016)
Natural aggregation algorithm	Luo et al. (2016)
Sunshine algorithm	Jahedbozorgan & Amjadifard (2016)
Virulence Optimization Algorithm	Jaderyan & Khotanlou (2016)
Crow Search Algorithm	Askarzadeh (2016)
Dragonfly algorithm	Mirjalili (2016)
Sperm whale algorithm	Ebrahimi & Khamehchi (2016)
Dynamic Virtual Bats Algorithm	Topal & Altun (2016)
Virus colony search	Li et al. (2016)
Monkey King Evolution	Meng & Pan (2016)
Bats sonar algorithm	Yahya et al. (2016)
Synergistic Fibroblast Optimization	Subashini et al. (2016)
Mosquito flying optimization	Alauddin (2016)
Whale Optimization Algorithm	Mirjalili & Lewis (2016)
Cyclical parthenogenesis algorithm	Kaveh & Zolghadr (2016)
Sperm motility algorithm	Abdel-Raouf & Hezam (2017)
Spotted Hyena Optimizer	Dhiman & Kumar (2017)

Biology-Based Algorithms (Evolutionary, Swarm intelligence, and Artificial Immune Systems)	
Algorithm	**Reference**
Grass Fibrous Root Optimization Algorithm	Akkar & Mahdi (2017)
Laying Chicken Algorithm	Hosseini (2017)
Grasshopper Optimisation Algorithm	Saremi et al. (2017)
Physics-Based Algorithms	
Simulated Annealing	Kirkpatrick et al. (1983)
Stochastic Diffusion Search	Bishop (1989)
Self-Propelled Particles	Vicsek et al. (1995)
Extremal Optimization	Boettcher & Percus (1999)
Intelligent Water Drops	Shah-Hosseini (2007)
Central Force Optimization	Formato (2007)
River Formation Dynamics	Rabanal et al. (2007)
Gravitational Search Algorithm	Rashedi et al. (2009)
Charged System Search	Kaveh & Talatahari (2010)
Galaxy-Based Search Algorithm	Shah-Hosseini (2011)
Spiral Optimization	Tamura & Yasuda (2011)
Electro-Magnetism Optimization	Cuevas et al. (2012)
Water Cycle Algorithm	Eskandar et al. (2012)
Big Bang-Big Crunch	Zandi et al. (2012)
Ray Optimization	Kaveh and Khayatazad (2012)
Black Hole Algorithm	Hatamlou (2012)
Mine blast algorithm	Sadollah (2012)
Colliding Bodies Optimization	Kaveh & Mahdavi (2014)
Vortex Search Algorithm	Doğan & Ölmez (2015)
Water Waves Optimization	Zheng (2015)
Optics Inspired Optimization	Kashan (2015)
Cloud Particles Evolution Algorithm	Li et al. (2015)
Heat transfer search	Patel & Savsani (2015)
Passing vehicle search	Savsani & Savsani (2015)
Lightning search Algorithm	Shareef et al. (2015)
Ions motion algorithm	Javidy et al. (2015)
Water Evaporation Optimization	Kaveh & Bakhshpoori (2016)
Drops Contact Optimization	Ghasemi-Ghalebahman and Moradi-Golestani (2016)
Multi-verse optimizer	Mirjalili et al. (2016)
Galactic Swarm Optimization	Muthiah-Nakarajan & Noel (2016)
Electromagnetic Field Optimization	Abedinpourshotorban (2016)
Thermal exchange optimization	Kaveh & Dadras (2017)
Sonar Inspired Optimization	Tzanetos & Dounias (2017)
Vibrating Particles System Algorithm	Kaveh & Ghazaan (2017)

Social-Based Algorithms	
Algorithm	**Reference**
Cultural algorithm	Reynolds (1994)
Grammatical Evolution	Ryan et al. (1998)
Imperialist Competitive Algorithm	Atashpaz-Gargari & Lucas (2007)
Social Emotional Optimization Algorithm	Xuet al. (2010)
Fireworks Optimization Algorithm	Tan & Zhu (2010)
Artificial Tribe Algorithm	Chen (2010)
Teaching–Learning-Based Optimization	Rao et al. (2011)
Brain Storm Optimization	Shi (2011)
Wisdom of Artificial Crowds	Yampolskiy et al. (2012)
Anarchic society optimization	Shayeghi & Dadashpour (2012)
Creativity-oriented optimization model	Feng et al. (2014)
Open source Development Model Algorithm	Khormouji et al. (2014)
Interior search algorithm	Gandomi (2014)
Simple Human Learning Optimization algorithm	Wang et al. (2014)
Exchange market algorithm	Ghorbani & Babaei (2014)
Artificial infectious disease optimization	Huang (2016)
Jigsaw Inspired Metaheuristic	Chifu et al. (2016)
Yin-Yang-pair Optimization	Punnathanam & Kotecha (2016)
Cohort intelligence algorithm	Kulkarni (2017)
Human mental search	Mousavirad & Ebrahimpour-Komleh, (2017)
Find-Fix-Finish-Exploit-Analyze Metaheuristic	Kashan et al. (2017)

Chemistry-Based Algorithms	
Algorithm	**Reference**
Artificial Chemical Process	Irizarry (2005)
Chemical Reaction Optimization	Lam and Li (2010)
Artificial Chemical Reaction Optimization	Alatas (2011)
Chemical Reaction Algorithm	Melin et al. (2013)
Gases Brownian Motion Optimization	Abdechiri et al. (2013)
Chemotherapy Science Algorithm	Salmani & Eshghi (2017)

Math-Based Algorithms	
Algorithm	**Reference**
Matheuristics	Boschetti et al. (2009)
Base Optimization Algorithm	Salem (2012)
Sine Cosine Algorithm	Mirjalili (2016)
Simulated Kalman Filter Algorithm	Ibrahim et al. (2016)
Golden Sine Algorithm	Tanyildizi & Demir (2017)

Sport-Based Algorithms	
Algorithm	**Reference**
Soccer League Competition algorithm	Moosavian & Roodsari (2013)
League Championship Algorithm	Kashan (2014)
Golden Ball	Osaba & Diaz (2014)
Football Game Algorithm	Fadakar (2016)
Tug of War Optimization	Kaveh & Zolghadr (2016)

Music-Based Algorithms	
Algorithm	**Reference**
Harmony Search	Geem et al. (2001)
Harmony Elements Algorithm	Cui et al. (2008)
Melody Search	Ashrafi & Dariane (2011)
Method of musical composition	Gutiérrez et al. (2012)

Appendix Non-Metaphor Based Algorithms	
Non-Metaphor Based Metaheuristics	
Algorithm	**Reference**
Scatter search	Glover (1977)
Tabu Search	Glover and McMillan (1986)
Guided Local Search	Voudouris and Tsang (1995)
Greedy Randomized Adaptive Search Procedures	Feo and Resende (1995)
Variable Neighborhood Search	Mladenović and Hansen (1997)
Cross Entropy Method	Rubinstein (1997)
Partial Optimization Metaheuristic Under Special Intensification Conditions	Taillard and Voss (2002)
Iterated Local Search Or Multi-Start Local Search	Lourenco et al. (2003)
Relaxation Induced Neighborhood Search	Danna et al. (2005)
Dialectic search	Kadioglu and Sellmann (2009)
Coalition-based metaheuristic	Meignan et al. (2010)
Global Simplex Optimization	Karimi & Siarry (2012)
Backtracking Optimization Search Algorithm	Civicioglu (2013)
Gradient Evolution Algorithm	Kuo & Zulvia (2015)
multi-wave algorithms	Glover (2016)
Sampling-Based Metaheuristic	Papapanagiotou et al. (2016)
Average-Based Design Optimization Algorithm	Cardoso & Barreiros (2017)
Fractal-based Algorithm	Kaedi (2017)
Global Sensitivity Analysis-Based Optimization Algorithm	Kaveh (2017)

REFERENCES

[1] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers & Operations Research 13 (5) (1986) 533–549.

[2] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Computing Surveys (CSUR) 35 (3) (2003) 268–308.

[3] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 67–82.

[4] Y.-C. Ho, D.L. Pepyne, Simple explanation of the no free lunch theorem of optimization, Cybernetics and Systems Analysis 38 (2) (2002) 292–298.

[5] I. Fister Jr, X.-S. Yang, I. Fister, J. Brest, D. Fister, A brief review of nature-inspired algorithms for optimization, arXiv:1307.4186, preprint.

[6] S. Binitha, S.S. Sathya, et al., A survey of bio inspired optimization algorithms, International Journal of Soft Computing and Engineering 2 (2) (2012) 137–151.

[7] X.-S. Yang, Nature-inspired algorithms: success and challenges, in: Engineering and Applied Sciences Optimization, Springer, 2015, pp. 129–143.

[8] A.K. Kar, Bio inspired computing–a review of algorithms and scope of applications, Expert Systems with Applications 59 (2016) 20–32.

[9] M. Khajehzadeh, M.R. Taha, A. El-Shafie, M. Eslami, A survey on meta-heuristic global optimization algorithms, Research Journal of Applied Sciences, Engineering and Technology 3 (6) (2011) 569–578.

[10] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Information Sciences 237 (2013) 82–117.

[11] K. Sörensen, F.W. Glover, Metaheuristics, in: Encyclopedia of Operations Research and Management Science, Springer, 2013, pp. 960–970.

[12] I.H. Osman, Focused issue on applied meta-heuristics, 2003.

[13] M. Gendreau, J.-Y. Potvin, Metaheuristics in combinatorial optimization, Annals of Operations Research 140 (1) (2005) 189–213.

[14] S. Akyol, B. Alatas, Plant intelligence based metaheuristic optimization algorithms, Artificial Intelligence Review 47 (4) (2017) 417–462.

[15] J.A. Ruiz-Vanoye, O. Díaz-Parra, F. Cocón, A. Soto, M. De los Ángeles, B. Arias, G. Verduzco-Reyes, R. Alberto-Lira, Meta-heuristics algorithms based on the grouping of animals by social behavior for the traveling salesman problem, International Journal of Combinatorial Optimization Problems and Informatics 3 (3) (2012) 104.

[16] L.J. Fogel, A.J. Owens, M.J. Walsh, Artificial intelligence through simulated evolution.

[17] I. Rechenberg, Evolution Strategy: Optimization of Technical Systems by Means of Biological Evolution, Fromman-Holzboog, Stuttgart 104.

[18] J.R. Sampson, Adaptation in natural and artificial systems, John h. Holland, 1976.

[19] J.R. Koza, Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems, Stanford University, Department of Computer Science, Stanford, CA, 1990.

[20] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, MHS'95, IEEE, 1995, pp. 39–43.

[21] M. Dorigo, Optimization, Learning and Natural Algorithms, Ph.D. Thesis, Politecnico di Milano, Italy, 1992.

[22] L.N. De Castro, J. Timmis, Artificial Immune Systems: A New Computational Intelligence Approach, Springer Science & Business Media, 2002.

[23] N. Nanas, A. De Roeck, Multimodal dynamic optimization: from evolutionary algorithms to artificial immune systems, in: Artificial Immune Systems, Springer, 2007, pp. 13–24.

[24] Z. Ji, D. Dasgupta, Revisiting negative selection algorithms, Evolutionary Computation 15 (2) (2007) 223–251.

[25] L.N. De Castro, F.J. Von Zuben, The clonal selection algorithm with engineering applications, in: Proceedings of GECCO, vol. 2000, 2000, pp. 36–39.

[26] L.N. De Castro, J. Timmis, An artificial immune network for multimodal function optimization, in: Proceedings of the 2002 Congress on Evolutionary Computation, 2002, vol. 1, CEC'02, IEEE, 2002, pp. 699–704.

[27] J. Kelsey, J. Timmis, Immune inspired somatic contiguous hypermutation for function optimisation, in: Genetic and Evolutionary Computation—GECCO 2003, Springer, 2003, p. 202.

[28] J.R. Al-Enezi, M.F. Abbod, S. Alsharhan, Artificial immune systems-models, algorithms and applications.

[29] K. Holzinger, V. Palade, R. Rabadan, A. Holzinger, Darwin or Lamarck? Future challenges in evolutionary algorithms for knowledge discovery and data mining, in: Interactive Knowledge Discovery and Data Mining in Biomedical Informatics, Springer, 2014, pp. 35–56.

[30] L.N. De Castro, Immune, swarm, and evolutionary algorithms. Part I: basic models, in: Proceedings of the 9th International Conference on Neural Information Processing, 2002, vol. 3, ICONIP'02, IEEE, 2002, pp. 1464–1468.
[31] L.N. de Castro, Immune, swarm, and evolutionary algorithms. Part II: philosophical comparisons, in: Proceedings of the 9th International Conference on Neural Information Processing, 2002, vol. 3, ICONIP'02, IEEE, 2002, pp. 1469–1473.
[32] A.S. Khalil, An investigation into optimization strategies of genetic algorithms and swarm intelligence, Artificial Life, 2001, portal of Swarm Intelligence at National Chin-Yi Institute of Technology in Taiwan.
[33] A.E. Eiben, E.H. Aarts, K.M. Van Hee, Global convergence of genetic algorithms: a Markov chain analysis, in: International Conference on Parallel Problem Solving from Nature, Springer, 1990, pp. 3–12.
[34] S.J. Louis, G.J. Rawlins, Predicting convergence time for genetic algorithms, Foundations of Genetic Algorithms 2 (1993) 141–161.
[35] W.A. Kosters, J.N. Kok, P. Floréen, Fourier analysis of genetic algorithms, Theoretical Computer Science 229 (1–2) (1999) 143–175.
[36] K. Sugihara, Measures for performance evaluation of genetic algorithms, in: Proc. 3rd Joint Conference on Information Sciences, 1997, pp. 172–175.
[37] Y. Zhou, Study on Genetic Algorithm Improvement and Application, Ph.D. thesis, Worcester Polytechnic Institute, 2006.
[38] D. Corus, D.-C. Dang, A.V. Eremeev, P.K. Lehre, Level-based analysis of genetic algorithms and other search processes, in: International Conference on Parallel Problem Solving from Nature, Springer, 2014, pp. 912–921.
[39] M. Pant, R. Thangaraj, A. Abraham, Particle swarm optimization: performance tuning and empirical analysis, Foundations of Computational Intelligence 3 (2009) 101–128.
[40] R. Agrawal, Implementation and performance analysis of particle swarm optimization algorithm using matlab, International Journal of Artificial Intelligence and Knowledge Discovery 1 (4) (2012) 6–11.
[41] S. Agarwal, A.P. Singh, N. Anand, Evaluation performance study of firefly algorithm, particle swarm optimization and artificial bee colony algorithm for non-linear mathematical optimization functions, in: 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), IEEE, 2013, pp. 1–8.
[42] E.A. Kaur, E.M. Kaur, A comprehensive survey of test functions for evaluating the performance of particle swarm optimization algorithm, International Journal of Hybrid Information Technology 8 (5) (2015) 97–104.
[43] Y. Zhang, S. Wang, G. Ji, A comprehensive survey on particle swarm optimization algorithm and its applications, Mathematical Problems in Engineering 2015 (2015) 1–38, https://doi.org/10.1155/2015/931256.
[44] Y.-J. Zheng, Water wave optimization: a new nature-inspired metaheuristic, Computers & Operations Research 55 (2015) 1–11.
[45] A.R. Mehrabian, C. Lucas, A novel numerical optimization algorithm inspired from weed colonization, Ecological Informatics 1 (4) (2006) 355–366.
[46] D. Simon, Biogeography-based optimization, IEEE Transactions on Evolutionary Computation 12 (6) (2008) 702–713.
[47] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Information Sciences 179 (13) (2009) 2232–2248.
[48] R. Oftadeh, M. Mahjoob, M. Shariatpanahi, A novel meta-heuristic optimization algorithm inspired by group hunting of animals: hunting search, Computers & Mathematics with Applications 60 (7) (2010) 2087–2098.
[49] X.-S. Yang, A new metaheuristic bat-inspired algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), 2010, pp. 65–74.
[50] B. Zhang, Y.-J. Zheng, Convergence analysis of water wave optimization algorithm, Computer Science 4 (2016) 009.
[51] S.F.M. Burnet, et al., The clonal selection theory of acquired immunity.
[52] A.Y. Lam, V.O. Li, Chemical-reaction-inspired metaheuristic for optimization, IEEE Transactions on Evolutionary Computation 14 (3) (2010) 381–399.
[53] D.T. Connolly, An improved annealing scheme for the qap, European Journal of Operational Research 46 (1) (1990) 93–100.
[54] E. Taillard, Fant: fast ant system.
[55] É. Taillard, Robust taboo search for the quadratic assignment problem, Parallel Computing 17 (4–5) (1991) 443–455.
[56] M. Abdechiri, M.R. Meybodi, H. Bahrami, Gases brownian motion optimization: an algorithm for optimization (GBMO), Applied Soft Computing 13 (5) (2013) 2932–2946.
[57] F. Lardeux, F. Saubion, J.-K. Hao, GASAT: a genetic local search algorithm for the satisfiability problem, Evolutionary Computation 14 (2) (2006) 223–253.
[58] L. Yingbiao, A genetic algorithm based on adapting clause weights, Chinese Journal of Computer 7 (2) (2005) 20–32.

[59] Z.W. Geem, J.H. Kim, G. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2) (2001) 60–68.
[60] Z.W. Geem, Music-Inspired Harmony Search Algorithm: Theory and Applications, vol. 191, Springer, 2009.
[61] A. Milad, Harmony search algorithm: strengths and weaknesses, Journal of Computer Engineering and Information Technology 2 (1) (2013) 1–7.
[62] I. Kougias, N. Theodosiou, A new music-inspired harmony based optimization algorithm. Theory and applications, in: International Conference on Protection and Restoration of the Environment X, 2010.
[63] O. Abdel-Raouf, M.A.-B. Metwally, A survey of harmony search algorithm, International Journal of Computer Applications 70 (28).
[64] D. Manjarres, I. Landa-Torres, S. Gil-Lopez, J. Del Ser, M.N. Bilbao, S. Salcedo-Sanz, Z.W. Geem, A survey on applications of the harmony search algorithm, Engineering Applications of Artificial Intelligence 26 (8) (2013) 1818–1831.
[65] D.G. Yoo, J.H. Kim, Z.W. Geem, Overview of harmony search algorithm and its applications in civil engineering, Evolutionary Intelligence 7 (1) (2014) 3–16.
[66] R.A. Mora-Gutiérrez, J. Ramírez-Rodríguez, E.A. Rincón-García, An optimization algorithm inspired by musical composition, Artificial Intelligence Review 41 (3) (2014) 301–315.
[67] R.A. Mora-Gutiérrez, J. Ramírez-Rodríguez, E.A. Rincón-García, A. Ponsich, O. Herrera, An optimization algorithm inspired by social creativity systems, Computing 94 (11) (2012) 887–914.
[68] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Applied Mathematics and Computation 188 (2) (2007) 1567–1579.
[69] M.G. Omran, M. Mahdavi, Global-best harmony search, Applied Mathematics and Computation 198 (2) (2008) 643–656.
[70] Q.-K. Pan, P.N. Suganthan, M.F. Tasgetiren, J.J. Liang, A self-adaptive global best harmony search algorithm for continuous optimization problems, Applied Mathematics and Computation 216 (3) (2010) 830–848.
[71] S.A. Salem, BOA: a novel optimization algorithm, in: 2012 International Conference on Engineering and Technology (ICET), IEEE, 2012, pp. 1–5.
[72] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26 (1) (1996) 29–41.
[73] S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, Knowledge-Based Systems 96 (2016) 120–133.
[74] X.-S. Yang, Flower pollination algorithm for global optimization, in: UCNC, Springer, 2012, pp. 240–249.
[75] X. Yang, Firefly algorithm, in: Nature-Inspired Metaheuristic Algorithms, Luniver Press (Chapter 8).
[76] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equation of state calculations by fast computing machines, The Journal of Chemical Physics 21 (6) (1953) 1087–1092.
[77] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, et al., Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[78] P.J. van Laarhoven, E.H. Aarts, Performance of the simulated annealing algorithm, in: Simulated Annealing: Theory and Applications, Springer, 1987, pp. 77–98.
[79] M. Nieto-Vesperinas, F. Fuentes, R. Navarro, Performance of a simulated-annealing algorithm for phase retrieval, JOSA A 5 (1) (1988) 30–38.
[80] C.R. Reeves, Modern Heuristic Techniques for Combinatorial Problems, John Wiley & Sons, Inc., 1993.
[81] X. Gu, The Behavior of Simulated Annealing in Stochastic Optimization, Iowa State University, 2008.
[82] H. Nezamabadi-Pour, F. Barani, Gravitational search algorithm: concepts, variants, and operators, in: Handbook of Research on Modern Optimization Algorithms and Applications in Engineering and Economics, IGI Global, 2016, pp. 700–750.
[83] R.V. Rao, V.J. Savsani, D. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Computer-Aided Design 43 (3) (2011) 303–315.
[84] E. Mezura-Montes, C.A.C. Coello, A simple multimembered evolution strategy to solve constrained optimization problems, IEEE Transactions on Evolutionary Computation 9 (1) (2005) 1–17.
[85] A.E. Muñoz Zavala, A.H. Aguirre, E.R. Villa, Diharce, Constrained optimization via particle evolutionary swarm optimization algorithm (peso), in: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, ACM, 2005, pp. 209–216.
[86] R.L. Becerra, C.A.C. Coello, Cultured differential evolution for constrained optimization, Computer Methods in Applied Mechanics and Engineering 195 (33) (2006) 4303–4322.
[87] F.-z. Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, Applied Mathematics and Computation 186 (1) (2007) 340–356.

[88] D. Karaboga, B. Basturk, Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems, Foundations of Fuzzy Logic and Soft Computing (2007) 789–798.
[89] A.H. Kashan, League championship algorithm (LCA): an algorithm for global optimization inspired by sport championships, Applied Soft Computing 16 (2014) 171–200.
[90] J.-J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer with local search, in: The 2005 IEEE Congress on Evolutionary Computation, 2005, vol. 1, IEEE, 2005, pp. 522–528.
[91] S.M. Abdulhamid, M.S.A. Latiff, S.H.H. Madni, O. Oluwafemi, A survey of league championship algorithm: prospects and challenges, arXiv preprint arXiv:1603.09728.
[92] F. Glover, C. McMillan, The general employee scheduling problem. An integration of MS and AI, Computers & Operations Research 13 (5) (1986) 563–573.
[93] F. Glover, E. Taillard, A user's guide to tabu search, Annals of Operations Research 41 (1) (1993) 1–28.
[94] F. Glover, Tabu Search Fundamentals and Uses, Graduate School of Business, University of Colorado, Boulder, 1995.
[95] M.M. Cangalovic, V. Kovacevic-Vujcic, L. Ivanovic, M. Drazic, M. Asic, Tabu search: a brief survey and some real-life applications, Yugoslav Journal of Operations Research 6 (1) (1996) 5–18.
[96] S. Hanafi, On the convergence of tabu search, Journal of Heuristics 7 (1) (2001) 47–58.
[97] M. Mastrolilli, L.M. Gambardella, Maximum satisfiability: how good are tabu search and plateau moves in the worst-case?, European Journal of Operational Research 166 (1) (2005) 63–76.
[98] F. Glover, M. Laguna, R. Marti, Principles of tabu search, Approximation Algorithms and Metaheuristics 23 (2007) 1–12.
[99] H. Pirim, E. Bayraktar, B. Eksioglu, Tabu search: a comparative study, in: Tabu Search, InTech, 2008.
[100] N. Mladenović, P. Hansen, Variable neighborhood search, Computers & Operations Research 24 (11) (1997) 1097–1100.
[101] J. Brownlee, Clever Algorithms: Nature-Inspired Programming Recipes, Jason Brownlee, 2011.
[102] P. Hansen, N. Mladenović, Variable neighborhood search: principles and applications, European Journal of Operational Research 130 (3) (2001) 449–467.
[103] P. Hansen, N. Mladenović, Developments of variable neighborhood search, in: Essays and Surveys in Metaheuristics, Springer, 2002, pp. 415–439.
[104] P. Hansen, N. Mladenovic, A Tutorial on Variable Neighborhood Search, Groupe d'études et de recherche en analyse des décisions, HEC Montréal, 2003.
[105] É.D. Taillard, S. Voss, Popmusic—partial optimization metaheuristic under special intensification conditions, in: Essays and Surveys in Metaheuristics, Springer, 2002, pp. 613–629.
[106] P. Shaw, Using constraint programming and local search methods to solve vehicle routing problems, in: International Conference on Principles and Practice of Constraint Programming, Springer, 1998, pp. 417–431.
[107] E. Taillard, La programmation a memoire adaptative et les algorithmes pseudo-gloutons: nouvelles perspectives pour les meta-heuristiques.
[108] R. Bent, P. Van Hentenryck, Spatial, temporal, and hybrid decompositions for large-scale vehicle routing with time windows, CP 6308 (2010) 99–113.
[109] T. Stützle, H.H. Hoos, Max–min ant system, Future Generation Computer Systems 16 (8) (2000) 889–914.
[110] T. Cai, F. Pan, J. Chen, Adaptive particle swarm optimization algorithm, in: Fifth World Congress on Intelligent Control and Automation, 2004, vol. 3, WCICA 2004, IEEE, 2004, pp. 2245–2247.
[111] H. Aguiar e Oliveira, L. Ingber, A. Petraglia, M.R. Petraglia, M.A.S. Machado, Stochastic Global Optimization and Its Applications with Fuzzy Adaptive Simulated Annealing, Springer Publishing Company, Incorporated, 2012.
[112] N.J. Cheung, X.-M. Ding, H.-B. Shen, Adaptive firefly algorithm: parameter analysis and its application, PloS ONE 9 (11) (2014) e112634.
[113] S. Walton, O. Hassan, K. Morgan, M. Brown, Modified cuckoo search: a new gradient free optimisation algorithm, Chaos, Solitons & Fractals 44 (9) (2011) 710–718.
[114] I.N. Trivedi, J. Pradeep, J. Narottam, K. Arvind, L. Dilip, Novel adaptive whale optimization algorithm for global optimization, Indian Journal of Science and Technology 9 (38) (2016) 1–6, https://doi.org/10.17485/ijst/2016/v9i38/101939.
[115] V. Kumar, D. Kumar, J.K. Chhabra, Improved grey wolf algorithm for optimization problems, in: International Symposium on "Fusion of Science & Technology", 2016, pp. 18–22.
[116] H.A. Hefny, S.S. Azab, Chaotic particle swarm optimization, in: 2010 The 7th International Conference on Informatics and Systems (INFOS), IEEE, 2010, pp. 1–8.
[117] B. Alatas, Chaotic harmony search algorithms, Applied Mathematics and Computation 216 (9) (2010) 2687–2699.
[118] P. Snaselova, F. Zboril, Genetic algorithm using theory of chaos, Procedia Computer Science 51 (2015) 316–325.

[119] H. Bingol, B. Alatas, Chaotic league championship algorithms, Arabian Journal for Science and Engineering 41 (12) (2016) 5123–5147.
[120] J. Kennedy, Bare bones particle swarms, in: Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003, SIS'03, IEEE, 2003, pp. 80–87.
[121] S.M. Farahani, A. Abshouri, B. Nasiri, M. Meybodi, A gaussian firefly algorithm, International Journal of Machine Learning and Computing 1 (5) (2011) 448.
[122] H. Wang, S. Rahnamayan, H. Sun, M.G. Omran, Gaussian bare-bones differential evolution, IEEE Transactions on Cybernetics 43 (2) (2013) 634–647.
[123] F. Zou, L. Wang, X. Hei, D. Chen, Q. Jiang, H. Li, Bare-bones teaching-learning-based optimization, The Scientific World Journal (2014).
[124] X.-S. Yang, S. Deb, S. Fong, Accelerated particle swarm optimization and support vector machine for business optimization and applications, Networked Digital Technologies (2011) 53–66.
[125] A. Ozkis, A. Babalik, Accelerated abc (a-abc) algorithm for continuous optimization problems, Lecture Notes on Software Engineering 1 (3) (2013) 262.
[126] M.N.M. Salleh, K. Hussain, Accelerated mine blast algorithm for anfis training for solving classification problems, International Journal of Software Engineering and Its Applications 10 (6) (2016) 161–168.
[127] M. Lohokare, B. Panigrahi, S. Pattanaik, S. Devi, A. Mohapatra, Optimal load dispatch using accelerated biogeography-based optimization, in: 2010 Joint International Conference on Power Electronics, Drives and Energy Systems (PEDES) & 2010 Power India, IEEE, 2010, pp. 1–5.
[128] D.F. Jones, S.K. Mirrazavi, M. Tamiz, Multi-objective meta-heuristics: an overview of the current state-of-the-art, European Journal of Operational Research 137 (1) (2002) 1–9.
[129] X. Gandibleux, M. Sevaux, K. Sörensen, V. T'kindt, Metaheuristics for Multiobjective Optimisation, vol. 535, Springer Science & Business Media, 2004.
[130] G.R. Zavala, A.J. Nebro, F. Luna, C.A.C. Coello, A survey of multi-objective metaheuristics applied to structural optimization, Structural and Multidisciplinary Optimization 49 (4) (2014) 537–558.
[131] N. Srinivas, K. Deb, Multiobjective optimization using nondominated sorting in genetic algorithms, Evolutionary Computation 2 (3) (1994) 221–248.
[132] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: nsga-ii, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.
[133] J. Moore, R. Chapman, Application of Particle Swarm to Multiobjective Optimization, Department of Computer Science and Software Engineering, Auburn University, 1999.
[134] M. Reyes-Sierra, C.C. Coello, Multi-objective particle swarm optimizers: a survey of the state-of-the-art, International Journal of Computational Intelligence Research 2 (3) (2006) 287–308.
[135] C.R. Raquel, P.C. Naval Jr, An effective use of crowding distance in multiobjective particle swarm optimization, in: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, ACM, 2005, pp. 257–264.
[136] M.A. Khanesar, M. Teshnehlab, M.A. Shoorehdeli, A novel binary particle swarm optimization, in: Mediterranean Conference on Control & Automation, 2007, MED'07, IEEE, 2007, pp. 1–6.
[137] M. Sayadi, R. Ramezanian, N. Ghaffari-Nasab, A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems, International Journal of Industrial Engineering Computations 1 (1) (2010) 1–10.
[138] Y. Sharafi, M.A. Khanesar, M. Teshnehlab, Discrete binary cat swarm optimization algorithm, in: 2013 3rd International Conference on Computer, Control & Communication (IC4), IEEE, 2013, pp. 1–6.
[139] M. Nemati, H. Momeni, N. Bazrkar, Binary black holes algorithm, International Journal of Computer Applications 79 (6).
[140] D. Rodrigues, X.-S. Yang, A.N. De Souza, J.P. Papa, Binary flower pollination algorithm and its application to feature selection, in: Recent Advances in Swarm Intelligence and Evolutionary Computation, Springer, 2015, pp. 85–100.
[141] J.C. Bean, Genetic algorithms and random keys for sequencing and optimization, ORSA Journal on Computing 6 (2) (1994) 154–160.
[142] M.-R. Chen, X. Li, N. Wang, H.-B. Xiao, An improved shuffled frog-leaping algorithm for job-shop scheduling problem, in: 2011 Second International Conference on Innovations in Bio-inspired Computing and Applications (IBICA), IEEE, 2011, pp. 203–206.
[143] F. Sayoti, M.E. Riffi, Random-keys golden ball algorithm for solving traveling salesman problem, International Review on Modelling and Simulations (IREMOS) 8 (1) (2015) 84–89.

[144] A. Ouaarab, B. Ahiod, X.-S. Yang, Random-key cuckoo search for the travelling salesman problem, Soft Computing 19 (4) (2015) 1099–1106.
[145] L. Zhang, Y. Chen, R. Sun, S. Jing, B. Yang, A task scheduling algorithm based on PSO for grid computing, International Journal of Computational Intelligence Research 4 (1) (2008) 37–43.
[146] E.A. Abdelhafiez, F.A. Alturki, A shaking optimization algorithm for solving job shop scheduling problem, Industrial Engineering and Management Systems 10 (1) (2011) 7–14.
[147] M.K. Marichelvam, T. Prabaharan, X.S. Yang, A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems, IEEE Transactions on Evolutionary Computation 18 (2) (2014) 301–305.
[148] K.W. Huang, J.L. Chen, C.S. Yang, A memetic gravitation search algorithm for solving permutation flow shop scheduling, in: Advanced Materials Research, vol. 1079, Trans Tech Publ, 2015, pp. 626–630.
[149] J. Krause, J. Cordeiro, R.S. Parpinelli, H.S. Lopes, A survey of swarm algorithms applied to discrete optimization problems, in: Swarm Intelligence and Bio-inspired Computation: Theory and Applications, Elsevier Science & Technology Books, 2013, pp. 169–191.
[150] A.H. Wright, et al., Genetic algorithms for real parameter optimization, Foundations of Genetic Algorithms 1 (1991) 205–218.
[151] A.Y. Lam, V.O. Li, J. James, Real-coded chemical reaction optimization, IEEE Transactions on Evolutionary Computation 16 (3) (2012) 339–353.
[152] G.R. Raidl, A unified view on hybrid metaheuristics, Hybrid Metaheuristics 4030 (2006) 1–12.
[153] C. Blum, A. Roli, Hybrid metaheuristics: an introduction, in: Hybrid Metaheuristics, Springer, 2008, pp. 1–30.
[154] M. Ehrgott, X. Gandibleux, Hybrid metaheuristics for multi-objective combinatorial optimization, in: Hybrid Metaheuristics, Springer, 2008, pp. 221–259.
[155] C. Blum, J. Puchinger, G. Raidl, A. Roli, et al., A brief survey on hybrid metaheuristics, in: Proceedings of BIOMA, 2010, pp. 3–18.
[156] L.-Y. Tseng, S.-C. Liang, A hybrid metaheuristic for the quadratic assignment problem, Computational Optimization and Applications 34 (1) (2006) 85–113.
[157] D. Binu, M. Selvi, A. George, MKF-Cuckoo: hybridization of cuckoo search and multiple kernel-based fuzzy c-means algorithm, AASRI Procedia 4 (2013) 243–249.
[158] X.-S. Yang, S. Deb, Cuckoo search via Lévy flights, in: World Congress on Nature & Biologically Inspired Computing, 2009, NaBIC 2009, IEEE, 2009, pp. 210–214.
[159] L. Chen, C.P. Chen, M. Lu, A multiple-kernel fuzzy c-means algorithm for image segmentation, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 41 (5) (2011) 1263–1274.
[160] E. Nabil, A modified flower pollination algorithm for global optimization, Expert Systems with Applications 57 (2016) 192–203.
[161] S.A.-F. Sayed, E. Nabil, A. Badr, A binary clonal flower pollination algorithm for feature selection, Pattern Recognition Letters 77 (2016) 21–27.
[162] T.G. Crainic, M. Toulouse, Parallel metaheuristics, in: Fleet Management and Logistics, Springer, 1998, pp. 205–251.
[163] G. Luque, E. Alba, Parallel Genetic Algorithms: Theory and Real World Applications, vol. 367, Springer, 2011.
[164] H. Narasimhan, Parallel artificial bee colony (pabc) algorithm, in: World Congress on Nature & Biologically Inspired Computing, 2009, NaBIC 2009, IEEE, 2009, pp. 306–311.
[165] C.C. Ribeiro, I. Rosseti, Efficient parallel cooperative implementations of grasp heuristics, Parallel Computing 33 (1) (2007) 21–35.
[166] E. Alba, Parallel Metaheuristics: A New Class of Algorithms, vol. 47, John Wiley & Sons, 2005.
[167] E. Alba, G. Luque, S. Nesmachnow, Parallel metaheuristics: recent advances and new trends, International Transactions in Operational Research 20 (1) (2013) 1–48.
[168] S. Rao, Engineering Optimization: Theory and Practice, 3rd edn., Wiley–Interscience, New York, 1996, New Age International (P) Limited Publishers, New Delhi, India, 2009.
[169] S. Mirjalili, S.M. Mirjalili, A. Hatamlou, Multi-verse optimizer: a nature-inspired algorithm for global optimization, Neural Computing and Applications 27 (2) (2016) 495–513.
[170] J.D. Gibbons, S. Chakraborti, Nonparametric statistical inference, in: International Encyclopedia of Statistical Science, Springer, 2011, pp. 977–979.
[171] P. Cowling, G. Kendall, E. Soubeiga, A hyperheuristic approach to scheduling a sales summit, in: International Conference on the Practice and Theory of Automated Timetabling, Springer, 2000, pp. 176–190.

[172] E.K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, R. Qu, Hyper-heuristics: a survey of the state of the art, Journal of the Operational Research Society 64 (12) (2013) 1695–1724.
[173] V.V. de Melo, W. Banzhaf, Drone squadron optimization: a novel self-adaptive algorithm for global numerical optimization, Neural Computing and Applications (2017) 1–28.
[174] J.P. Papa, G.H. Rosa, D. Rodrigues, X.-S. Yang, Libopt: an open-source platform for fast prototyping soft optimization techniques, arXiv preprint arXiv:1704.05174.
[175] K. Sörensen, Metaheuristics—the metaphor exposed, International Transactions in Operational Research 22 (1) (2015) 3–18.