Computers & Industrial Engineering 137 (2019) 106040

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A survey on new generation metaheuristic algorithms

Tansel Dokeroglu[a,*], Ender Sevinc[b], Tayfun Kucukyilmaz[a], Ahmet Cosar[b]

[a] TED University, Computer Engineering Department, Ankara, Turkey
[b] University of THK, Computer Engineering Department, Ankara, Turkey

ARTICLE INFO

Keywords:
Survey
Metaheuristic
Algorithm
Optimization

ABSTRACT

Metaheuristics are an impressive area of research with extremely important improvements in the solution of intractable optimization problems. Major advances have been made since the first metaheuristic was proposed and numerous new algorithms are still being proposed every day. There is no doubt that the studies in this field will continue to develop in the near future. However, there is an obvious demand to pick out the best performing metaheuristics that are expected to be permanent. In this survey, we distinguish fourteen new and outstanding metaheuristics that have been introduced for the last twenty years (between 2000 and 2020) other than the classical ones such as genetic, particle swarm, and tabu search. The metaheuristics are selected due to their efficient performance, high number of citations, specific evolutionary operators, interesting interaction mechanisms between individuals, parameter tuning/handling concepts, and stagnation prevention methods. After giving absolute foundations of the new generation metaheuristics, recent research trends, hybrid metaheuristics, the lack of theoretical foundations, open problems, advances in parallel metaheuristics and new research opportunities are investigated.

1. Introduction

The term *metaheuristic* describes higher level heuristics that are proposed for the solution a wide range of optimization problems. Recently, many metaheuristics algorithms are successfully being applied for solving intractable problems. The appeal of using these algorithms for solving complex problems is that they obtain the best/optimal solutions even for very large problem sizes in small amounts of time.

The optimization problems that attracted the attention of metaheuristic approaches have a large variance, ranging from single to multiobjective, continuous to discrete, constrained to unconstrained. Solving these problems is not a straightforward task due to their complex behavior. Exact algorithms are mostly non-polynomial and, although providing best solutions usually have impractical execution times and/or computational requirements for large data sizes. Metaheuristic algorithms provide a practical and elegant solution to many such problems and are designed to achieve approximate/optimal solutions in practical execution times for NP-Hard optimization problems (Neumann & Witt, 2010).

The majority of the state-of-the-art metaheuristics have been developed before the year 2000. We name these algorithms as "*classical*" metaheuristic algorithms in this survey. The aforementioned classical algorithms are: Genetic Algorithms (GA) (Goldberg, 1989), Particle Swarm Optimization (PSO) (Wei & Qiqiang, 2004), Ant Colony Optimization (ACO) (Dorigo & Birattari, 2010), Genetic Programming (GP) (Banzhaf, Nordin, Keller, & Francone, 1998), Differential Evolution (DE) (Storn & Price, 1997), Simulated Annealing (SA) (Van Laarhoven & Aarts, 1987), Tabu search (TS) (Glover & Laguna, 1998), Greedy Randomized Adaptive Search Procedure (GRASP) (Marques-Silva & Sakallah, 1999), Artificial Immune Algorithm (AIA) (Dasgupta, 2012), Iterated Local Search (ILS) (Lourenço, Martin, & Stützle, 2003), Chaos Optimization Method (COM) (Li & Jiang, 1997), Scatter Search (SS) (Martí, Laguna, & Glover, 2006), Shuffled Frog-Leaping Algorithm (SFLA) (Eusuff & Lansey, 2003), and Variable Neighborhood Search (VNS) (Mladenović & Hansen, 1997).

Despite the achievements of the classical metaheuristic algorithms, new and novel evolutionary approaches also emerged successfully in the last two decades. Research on metaheuristic algorithms during this era introduces a great number of new metaheuristics inspired by evolutionary or behavioral processes. In many instances, this new wave of metaheuristic approaches yield the best solutions for some of the unsolved benchmark problem sets.

In this survey, we review the last twenty years of metaheuristic algorithms. Due to the worldwide popularity and success of the studies on metaheuristics and the increasing publication counts of these

* Corresponding author.
E-mail address: tansel.dokeroglu@tedu.edu.tr (T. Dokeroglu).

https://doi.org/10.1016/j.cie.2019.106040
Received 25 May 2019; Received in revised form 29 August 2019; Accepted 2 September 2019
Available online 06 September 2019
0360-8352/ © 2019 Elsevier Ltd. All rights reserved.

studies, we consider that there is a need for a new survey to review and summarize the most appealing current studies on this subject for the last 20 years. We have concentrated our efforts on studying fourteen distinguished metaheuristic algorithms that we term as the "*new generation*" metaheuristic algorithms. While selecting/deciding the new generation metaheuristics, we focus on the number of citations that have been received with respect to the introduction year of the metaheuristic. Therefore, the metaheuristic becomes verified to be efficient by the experimental studies of a large number of scientists. The selected metaheuristic should also introduce a novelty in one of the issues, operators for exploration and exploitation techniques, parameter tuning/reduction concepts, adaptability to the solution of a wide range of problems, and stagnation prevention techniques. The metaheuristic must have been also reported to outperform some of the classical metaheuristics.

The new generation metaheuristic algorithms that we examine in this review are: Artificial Bee Colony (ABC) (Karaboga, 2005), Bacterial Foraging (BFO) (Das, Biswas, Dasgupta, & Abraham, 2009), Bat Algorithm (BA) (Yang, 2010c), Biogeography-based optimization (BFO) (Simon, 2008), Cuckoo Search (CS) (Yang & Deb, 2009), Firefly Algorithm (FA) (Yang, 2010a), Gravitational Search Algorithm (GSA) (Rashedi, Nezamabadi-Pour, & Saryazdi, 2009), Grey Wolf Algorithm (GWA) (Mirjalili, Mirjalili, & Lewis, 2014), Harmony Search (HS) (Geem, Kim, & Loganathan, 2001), Krill Herd (KH) (Gandomi & Alavi, 2012), Social Spider Optimization (SSO) (Cuevas, Cienfuegos, ZaldíVar, & Pérez-Cisneros, 2013), Symbiotic Organisms Search (SOS) (Cheng & Prayogo, 2014), Teaching Learning Based Optimization (TLBO) (Rao, Savsani, & Vakharia, 2011), and Whale Optimization Algorithm (WOA) (Mirjalili & Lewis, 2016).

Figs. 1 and 2 give the search result of the number of related studies for the classical and new generation metaheuristics on google scholar website (in May 2019). GA and ABC have the largest numbers 1,270,000 and 37,400 related papers respectively.

The rest of this survey is organized as follows: previous surveys on classical metaheuristics are presented in Section 2. Section 3 gives the details and pseudocode of new generation metaheuristics. Section 4 gives information about other recent metaheuristic algorithms that are not known as much as the presented metaheuristics in the previous section. Section 5 provides information about other new generation hybrid metaheuristics. In Section 6, a comprehensive discussion and concluding remarks are provided.

2. Previous surveys on classical metaheuristics

This section aims to provide concise information about the previous surveys on classical metaheuristic algorithms. Here, we have selected

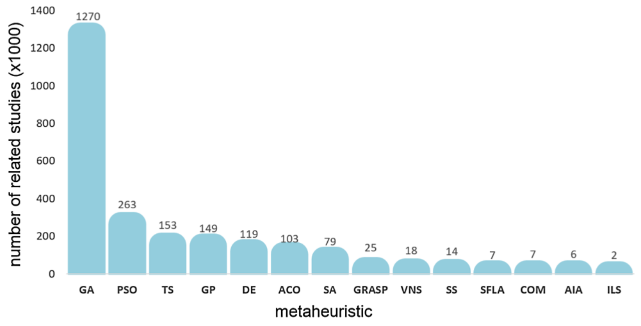

Fig. 1. The number of related papers on google scholar for the classical metaheuristics.

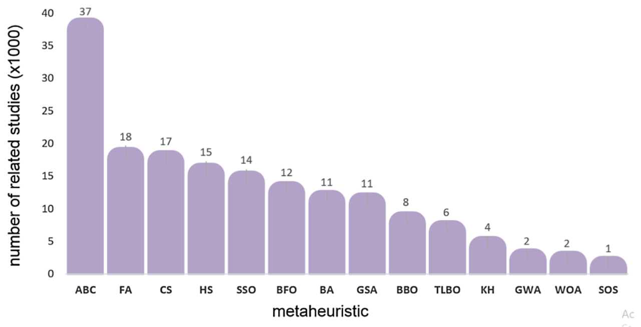

Fig. 2. The number of related papers on google scholar for new generation metaheuristics summarized in this survey.

reviews that have a remarkable impact on recent studies reflected via citation counts. While acting as an introduction to the metaheuristic approaches concentrated in this study, this section also provides a comprehensive reference for readers that are interested in classical metaheuristic algorithms.

The book by Goldberg (1989), GA in Search Optimization & Machine Learning, is one of the first publications in the field of metaheuristics designed by natural inspirations. Goldberg and Holland (1988) provide a book dedicated to the papers related to GA and genetics-based learning systems with a focus on machine learning. Holland (1992) presents brief information on the fundamental properties of GA in his study. Srinivas and Patnaik (1994) provide a survey on the basic operators (crossover and mutation) of GA and introduce research opportunities for complex problem landscapes. Schaffer, Whitley, and Eshelman (1992) provide a survey on GA and the neural networks. The book, "Nature-inspired metaheuristic algorithms" by (Yang, 2010b), is a comprehensive reference for the state-of-the-art metaheuristic algorithms. The book offers a review of state-of-the-art metaheuristics introduced before 2010. BoussaïD, Lepagnot, and Siarry (2013) present a survey on some main metaheuristics and examine their similarities and differences. The authors classify the methods as a single solution and population-based metaheuristics and present an overall evaluation of the main metaheuristics and their principles. Sörensen, Sevaux, and Glover (2018) describe his comments on the new proposed metaheuristics and the similarities that occur in many.

Wei and Qiqiang (2004) summarize the basic principles of PSO algorithm that is introduced as a new optimization algorithm originated from artificial life. The PSO finds the best/optimal solutions through improving the global and the local best solutions of particles in the population. The parameters of the PSO are examined in this study and the areas that the PSO has been applied are reviewed. Taillard, Gambardella, Gendreau, and Potvin (2001) examine memory-based metaheuristics TS, SS, GA and ACO in their survey. They propose that their implementations are similar and should be unified with the name of Adaptive Memory Programming. Bianchi, Dorigo, Gambardella, and Gutjahr (2009) give a survey of metaheuristic algorithms such as ACO, TS, SA, and evolutionary computation. The success of these applications for the class of Stochastic Combinatorial Optimization Problems is analyzed and recent issues are discussed. Parejo, Ruiz-Cortés, Lozano, and Fernandez (2012) present a comparative study on metaheuristic optimization frameworks. Diverse metaheuristic techniques, for solution encoding, constraints, neighborhood, hybrid solutions, parallel/distributed computation, best practices of software engineering, and documentation are covered. A significant lack of implementation is reported for the parallel computation of metaheuristics and hyperheuristics. Mladenović, Brimberg, Hansen, and Moreno-Pérez (2007) provide a survey of metaheuristics for the solution of the *p*-

median problem. The authors give an overview of metaheuristic algorithms for this problem. Puchinger and Raidl (2005) discuss different state-of-the-art techniques to combine metaheuristics with brute-force algorithms to optimize combinatorial problems. They report two categories as collaborative versus integrative combinations. de Castro and Timmis (2003) propose a framework for AIS and review literature that integrates AIS with other algorithms, neural networks, evolutionary computation, and fuzzy systems.

Li and Jiang (1997) introduce a chaos optimization algorithm that uses the properties of stochastic property, the regularity of chaos, and ergodicity. The performance of the chaos optimization algorithm is reported to be very high. Espejo, Ventura, and Herrera (2010) provide a survey of GP for the classification problems. GP is reported to be a powerful evolutionary technique that is suitable for the evolution of classifiers. The study surveys the literature to give information about the techniques of constructing well-performing classifiers. Lewis (2008) carries out an overview of metaheuristic algorithms belonging to the university timetabling problems. The author classifies the algorithms into three general classes, and comment on them. Nanda and Panda (2014) provide a review of nature-inspired algorithms for the problem of partitional clustering. Key issues and major practice areas are investigated. Zavala, Nebro, Luna, and Coello (2014) examine recent developments in multiobjective metaheuristics for solving design problems of civil engineering structures. The authors examine the design problems and the features of the problem-solving methods. Baghel, Agrawal, and Silakari (2012) give a review of combinatorial optimization problems that are solved by metaheuristics. The paper examines the evolution of metaheuristic and their process of converging. The authors divide the metaheuristic algorithms categories and make some suggestions to develop well-performed metaheuristic algorithms.

Bianchi, Dorigo, Gambardella, and Gutjahr (2006) introduce ACO, Evolutionary Computation, SA, TS and Stochastic Partitioning methods and their recent applications in their survey. The authors mention the flexibility of metaheuristics in adapting to different modeling approaches. A description and classification of the modeling approaches of optimization under uncertainty are provided. Blum, Puchinger, Raidl, and Roli (2010) give a survey on the approaches of hybrid metaheuristics. They mention that combining different metaheuristic algorithms is one of the most successful techniques in optimization. Dorigo and Blum (2005) provide a survey on theoretical notations of ACO. The authors examine the convergence outcomes and discuss the relations of ACO and other optimization methods. They focus on research efforts of understanding the behavior of ACO. Feature aspects of ACO are given in this study. Mohan and Baskaran (2012) review recent research ACO algorithms and propose a new ACO algorithm that is applied for network routing. Hao and Solnon (2019) describe the common features of metaheuristics by grouping them in two main approaches, perturbative and constructive metaheuristics. The authors also introduce the diversification and intensification notions, which are used by metaheuristics.

Pedemonte, Nesmachnow, and Cancela (2011) present a comprehensive survey on parallel ACO implementations and give a new taxonomy to classify parallel ACO algorithms. Cantú-Paz (1998) provides detailed information about parallel GA. The author collects and presents the most representative publications on parallel GA. Blum, Puchinger, Raidl, and Roli (2011) provide a survey on hybrid metaheuristic algorithms with other optimization techniques. They define the research area with different hybridization methods. They propose some methods to develop efficient hybrid metaheuristic algorithms. Alba (2005) gives a comprehensive survey on parallel GA (Alba & Troya, 1999). The author defines parallel GAs as a new kind of metaheuristics. In addition to parallel GA, the parallel versions of GP, SS, ACO, SA, VNS, TS, hybrid, heterogeneous, and multiobjective are evaluated. The theoretical foundations are analyzed. The parallel algorithms are reported to be highly efficient and robust. Mühlenbein (1992) shows the power of a Parallel GA with the *m* graph partitioning and the traveling salesman problems. The parallel GA are reported to be capable of finding the best and the optimal solutions for the problem instances. Alba, Luque, and Nesmachnow (2013) provide a solution to deal with classic parallel models in contemporary platforms. They review modern research areas with respect to parallel metaheuristics and identify possible open research areas and future trends. Binitha et al. (2012) present an overview of biologically inspired metaheuristic algorithms. Del Ser et al. (2019) survey the state-of-the-art metaheuristics and report open research areas. A discussion is carried out on recent studies and identifies the necessity to reach a common understanding of optimization techniques. Camacho-Villalón, Dorigo, and Stützle (2019) analyze the Intelligent Water Drops (IWD) metaheuristic and verify that the IWD is inspired by ACO metaheuristic.

In the fields of supply chain and inventory management optimization, metaheuristic algorithms are used intensively. For detailed information, the reader can refer to the studies (Awasthi & Omrani, 2019; Duan, Deng, Gharaei, Wu, & Wang, 2018; Dubey, Gunasekaran, & Sushil Singh, 2015; Gharaei, Hoseini Shekarabi, & Karimi, 2019; Gharaei, Karimi, & Hoseini Shekarabi, 2019; Gharaei, Karimi, & Shekarabi, 2019; Giri & Bardhan, 2014; Giri & Masanta, 2018; Hao, Helo, & Shamsuzzoha, 2018; Hoseini Shekarabi, Gharaei, & Karimi, 2019; Kazemi, Abdul-Rashid, Ghazilla, Shekarian, & Zanoni, 2018; Rabbani, Foroozesh, Mousavi, & Farrokhi-Asl, 2019; Rabbani, Hosseini-Mokhallesun, Ordibazar, & Farrokhi-Asl, 2018; Sarkar & Giri, 2018; Sayyadi & Awasthi, 2018a, 2018b; Shah, Chaudhari, & Cárdenas-Barrón, 2018; Tsao, 2015; Yin, Nishi, & Zhang, 2016).

3. New generation metaheuristics

In this section, we provide comprehensive information for fourteen recent metaheuristics that have attracted the attention of many researchers and have been cited numerous times for the last twenty years. All of these algorithms are nature or human inspired, and population-based. The algorithms are presented in alphabetical order. For each of the algorithms, we reserve a specific section, providing details about their origins, inspirations, pseudocode-level details, and standing research on them.

3.1. Artificial bee colony optimization (ABC)

ABC is a metaheuristic algorithm proposed by Karaboga in 2005. It is one of the most cited new generation metaheuristics in literature (Basturk, 2006; Karaboga, 2005; Karaboga, Gorkemli, Ozturk, & Karaboga, 2014). ABC, being a population-based algorithm, has been applied to various optimization problems. The natural aspiration of ABC comes from the fact that candidate solutions are represented as bees exploring/exploiting food resources, and solutions are represented as the food resources themselves (Dokeroglu, Sevinc, & Cosar, 2019). A solution indicates a food resource and the nectar amount of each resource represents the quality/fitness of each solution. There are three types of bees in the hive: "*employed*", "*onlooker*", and "*scout*" bees. In nature, employed bees look for a food source, come back to hive and share their information by dancing. When an employed bee finishes the collection of the nectar, it turns into a scout and looks for new food resources. Onlooker bees watch how the employed bees dance and choose food sources, while scout bees explore for food sources. Computationally, the bee colony and its behavior are represented as follows (see Table 1).

First, a random initial population is generated. The fitness of a state of a bee colony is indicated by the acquired resources. Fig. 3 presents

Table 1
Summary information about the new generation metaheuristics, their inventors and the year they are introduced.

#	Acronym	Metaheuristic	Inventors, Year
1	HS	Harmony Search	Geem et al. (2001)
2	BFO	Bacterial Foraging Opt.	Passino (2002)
3	ABC	Artificial Bee Colony	Karaboga (2005)
4	BBO	Biogeography-based Opt.	Simon (2008)
5	CS	Cuckoo Search	Yang and Deb (2009)
6	GSA	Gravitational Search Algorithm	Rashedi et al. (2009)
7	FA	Firefly Algorithm	Yang (2010a)
8	BA	Bat Algorithm	Yang (2010c)
9	TLBO	Teaching-learning-Based Opt.	Rao et al. (2011)
10	KH	Krill Herd	Gandomi and Alavi (2012)
11	SSO	Social spider optimization	Cuevas et al. (2013)
12	GWA	Grey Wolf Algorithm	Mirjalili, Mirjalili, and Lewis (2014)
13	SOS	Symbiotic Organisms Search	Cheng and Prayogo (2014)
14	WOA	Whale Optimization	Mirjalili and Lewis (2016)

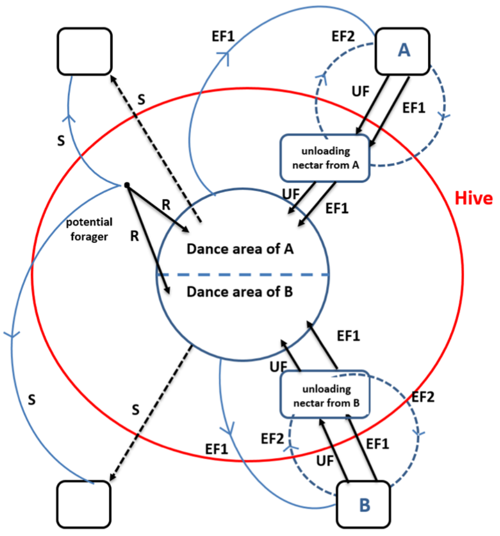

Fig. 3. The classical behavior of honeybees looking for nectar.

the basic behavior of artificial bees. A forager bee starts as an unemployed bee having no information about the food sources around the hive. An ordinary bee can be a scout bee and explore the solution space (see S in Fig. 3) or it can watch the dance of other bees and search for new food sources, R. The bee gathers the food, comes back to the hive, drops off the nectar. The bee can become a recruit nestmates (EF1), an uncommitted follower (UF), or go searching the food without recruiting after bees (EF2). The pseudocode of an ABC algorithm is given in Algorithm 1.

In order to generate an initial set of food sources, the algorithm starts randomly generating solutions in the search space. The Equation given below defines the randomized rule to produce a new solution within the range of the boundaries of the parameters. $i = 1 \ldots SN$, $j = 1 \ldots D$. SN is the size of the food sources and D is the number of dimensions in the problem space. X_{ij} is the j-th dimension of food source i.

$$X_{ij} = X_j^{min} + rand(0, 1)(X_j^{max} - X_j^{min}) \tag{1}$$

Each employed bee produces a new food source (solution) depending on its local information and finds a neighboring food source. Finding a neighboring food source is defined in the Equation given below.

$$V_{ij} = X_{ij} + \phi_{ij}(X_{ij} - X_{kj}) \tag{2}$$

A new food source V_i is produced by changing one parameter of x_i. In the Equation above, j is a random integer between $[1, D]$ where k is a random index different from i and ϕ_{ij} is a real number between $[-1, 1]$. As the difference between X_{ij} and X_{kj} diminishes, the perturbation on solution X_{ij} becomes smaller. If a new generated parameter exceeds its boundaries, the parameter is set to acceptable values. Fitness value of the new generated solution is calculated as $(1/(1 + f_i))$ when f_i is positive. The fitness value is $(1/(1 + abs(f_i)))$ if f_i is positive negative.

The composition of the bee colony impacts the balance between the exploration/exploitation of a run of ABC. Karaboga et al. compare the performance of ABC against GA, and PSO (Karaboga & Basturk, 2007). ABC algorithm is observed to outperform the algorithms on the optimization of multi-variable functions. Gao and Liu (2012) state that there is an insufficiency of ABC in its solution search equation at the exploitation phase and propose an improved fitness equation (inspired by DE) based on the bee search behavior around the best nectar in the previous iterations. The experimental results indicate that the modified ABC algorithm performs well, solving complex numerical problems compared to classical ABC algorithms.

Algorithm 1. Artificial Bee Colony Optimization (Karaboga, 2005)

```
1  int i=0;
2  while (i++ < #iterations) do
3      Scout bees search for food();
4      Scout bees return to the hive and dance();
5      Onlooker bees evaluate the food sources();
6      Check previously visited food resources();
7      Decide the best food resources();
8      Employed bees travel to the food sources();
9      Return to hive();
10     Collect the solution in the hive();
```

Karaboga and Ozturk (2011) propose an ABC algorithm for data clustering problem. The ABC algorithm is compared with other metaheuristic algorithms in the literature. The UCI Machine Learning Repository is examined during the experiments. The results verify that the ABC can be used for multivariate data clustering problems efficiently. Zhu and Kwong (2010) propose an ABC algorithm by using the knowledge of global best solution during the exploitation. The experimental results show that proposed GABC algorithm outperforms the classical ABC algorithm. The authors aim to solve the insufficiency of ABC regarding its solution search equation.

Karaboga and Basturk (2008) compare the performance of ABC algorithm with DE, PSO and GA for multi-dimensional numeric problems. The results obtained from their experiments verify that the ABC algorithm is competitive with other algorithms in the literature. The ABC can be efficiently used to solve high dimensional engineering problems (Karaboga & Basturk, 2008). Hancer, Xue, Zhang, Karaboga, and Akay (2018) propose a multiobjective ABC algorithm combined with non-dominated sorting and genetic operators for feature subset selection problem. Two different ABC versions are developed (with binary and continuous representation). Omkar, Senthilnath, Khandelwal, Naik, and Gopalakrishnan (2011) propose a modified ABC algorithm for discrete variables of the multiobjective design optimization of composite structures.

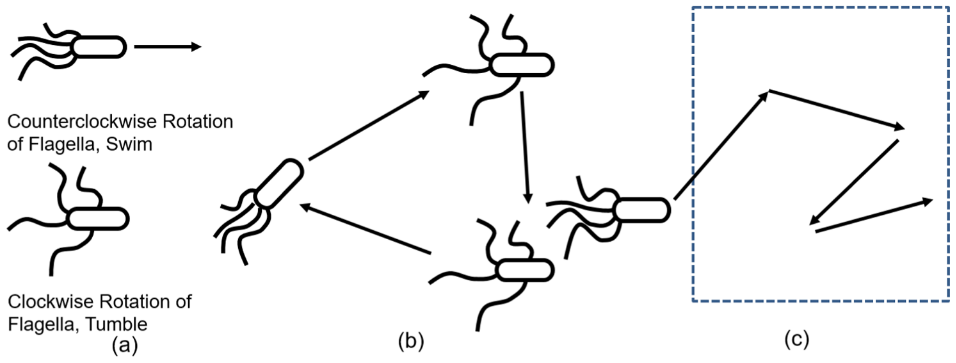
Counterclockwise Rotation of Flagella, Swim
Clockwise Rotation of Flagella, Tumble
(a)
(b)
(c)

Fig. 4. Swimming, tumbling, and chemotactic behavior of *E. coli*.

Karaboga (2009) proposes a new ABC algorithm for designing digital Infinite Impulse Response filters. The algorithm is compared with a conventional optimization algorithm PSO. Singh (2009) proposes an ABC method for the solution of minimum spanning tree problem with leaf-constrains. The experimental results verify the superiority of the proposed ABC algorithm. TSai, Pan, Liao, and Chu (2009) propose a new ABC optimization algorithm for numerical optimization. The algorithm introduces the universal gravitation concept into the affection consideration between the onlooker and employed bees in order to improve the quality of the solutions via selecting a more suitable exploration/exploitation ratio.

3.2. Bacterial foraging optimization (BFO)

Passino (2002) proposes BFO metahuristic. The BFO algorithm imitates the foraging behavior of bacteria over a landscape to process parallel non-gradient optimization. The movement (locomotion) is provided by a set of tensile flagella that helps an *E. coli* bacterium to swim or tumble during the foraging process. Each flagellum tightens the cell while they are turning around the flagella in a clockwise direction. This causes flagella to behave independently and the bacterium tumbles with lesser number of tumbling. In a detrimental location, it tumbles drastically to obtain a nutrient gradient. Turning the flagella counter clockwise helps the bacterium to swim quickly. The bacteria may face with chemotaxis, where they intend to move to a nutrient gradient and avoid a poisonous environment. The bacteria can move longer distances in a friendly environment. Fig. 4 presents the movement of a bacterium in a nutrient solution.

When the bacteria get enough food, they can generate a replica of itself. Passino is inspired by this event while developing the BFO algorithm. The chemotactic progress may not occur due to sudden changes in the environment. A group of bacteria may migrate to some other places or other groups of bacteria may come to the current location of the bacteria. This process is denoted as elimination-dispersal in which the bacteria in a region are terminated or a group of bacteria is dispersed into a new location in the environment.

If we assume that Θ is a vector of multidimensional real values where $J(\Theta)$ is the optimization problem, the BFOA uses chemotaxis, swarming, reproduction, and elimination-dispersal to optimize a given combinatorial problem. Virtual bacteria are implemented to locate the global optimum solution while dealing with the problem. Here, chemotoxis, reproduction, and elimination-dispersal steps facilitate the exploration process of BFO while, swarming behavior and reproduction facilitate the exploration process of the algorithm.

The structures used to define chemotactic steps are: S is #bacteria in population, k is the index for the reproduction step, j is the index for chemotactic step, p is the dimension of problem, l is the index of the elimination-dispersal event, N_c is #chemotactic steps, N_s is the length of swimming, N_{re} is #reproduction steps, N_{ed} is #elimination-dispersal events, P_{ed} is the probability of elimination-dispersal, and $C(i)$ is the size of the step taken in the random direction specified by the tumble.

Let $P(j,k,l) = \{\Theta^i\ (j,k,l) \mid 1,2,\ldots,S\}$ denote the position of each individual S bacteria at the j-th chemotactic step, k-th reproduction step, and l-th elimination-dispersal event. Here, $J(i, j, k, l)$ denotes the cost at the location of the i-th bacterium at location $\Theta^i\ (j, k, l)$. The four main components of BFOA are;

Chemotaxis: simulates the movement of an *E. coli* by swimming and tumbling using flagella. The movement of the bacterium can be represented as below:

$$\Theta^i\left(j+1, k, l\right) = \Theta^i\left(j, k, l\right) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\cdot\Delta(i)}} \tag{3}$$

where Δ is a vector in a random direction with elements between $[-1, 1]$.

Swarming: A swarm behavior is observed for motile species where they form intricate and stable spatio-temporal patterns in semisolid nutrient medium. A swarm of *E. coli* cells shows themselves in a ring by moving up the nutrient gradient when located among a semisolid matrix with a single nutrient chemo-effecter. The cell-to-cell communication formulation in *E. coli* group is represented by the function below:

$$J_{cc}\left(\Theta, P\left(j, k, l\right)\right) = \sum_{i=1}^{S} J_{cc}\left(\Theta, \Theta^i\left(j, k, l\right)\right) \tag{4}$$

where $J_{cc}(\Theta, P(j, k, l)$ is the value to be added to the objective function.

Reproduction: The unhealthy bacteria terminate while each of the healthy bacteria is being split into two bacteria. This process maintains the size of the swarm constant. The selection criteria for unhealthy bacteria is important for both exploration and exploitation utility of BFO.

Elimination and dispersal: Changes in the environment where bacteria lives may happen due to many reasons. A local temperature rise may terminate a group of bacteria in a region. Events can happen in this way and all the bacteria in a region may be killed or a group may migrate into a new location. In BFOA, some bacteria are liquefied to simulate this phenomenon randomly, while new substitutions are initiated in the search field. The pseudocode of the BFOA is presented in Algorithm 2.

Algorithm 2. Bacterial Foraging Optimization Algorithm (Das et al., 2009)

```
1  Initialize parameters $p$, $S$, $N_c$,$N_s$, $N_{re}$, $N_{ed}$, $P_{ed}$, $C(i)$($i$=1,2,...,S), $\Theta^i$;
2  //Elimination-dispersal loop
3  for ($l$=1 to $N_{ed}$) do
4      //Reproduction loop
5      for ($k$=1 to $N_{re}$) do
6          //Chemotaxis loop
7          for ($j$=1 to $N_c$) do
8              //bacterium i $\in$ Population
9              for ($i$=1,2,...S) do
10                 Compute fitness function, $J(i, j, k, l)$;
11                 $J_{last}$=$J(i, j, k, l)$ ;
12                 //hold the value since a better cost can be found
13                 Tumble: generate a random $\triangle(i) \in R^p$ with each element
                    $\triangle_m(i)$, $m$=1,2,...,$p$ on [-1,1];
14                 Move: $\Theta^i(j+1, k, l)$;
15                 //$C(i)$ in the direction of the tumble for bacterium $i$
16                 Compute $J(i$, $j$+1, $k$, l);
17                 while ($m < N_s$) do
18                     //Swim
19                     if ($J(i,j+1,k,l) < J_{last}$) then
20                         $J_{last} = J(i,j+1,k,l)$ ;
21                         $\Theta^i(j+1,k,l) = \Theta^i(j,k,l) + C(i)\frac{\triangle(i)}{\sqrt{\triangle^T(i).\triangle(i)}}$;
22                     else
23                         $m$=$N_s$;
24                     $m$=$m$+1;
```

Passino (2010) provides a tutorial about BFO, which summarizes the biology of bacterial foraging. Das et al. (2009) discuss the hybrid BFO algorithms with other optimization techniques and give information about the most significant applications of BFO. Dasgupta, Das, Abraham, and Biswas (2009) introduce new methods modifying how an individual and groups of bacteria forage for nutrients. The introduced model is designed for distributed optimization processes. Agrawal, Sharma, and Bansal (2012) present a survey on BFO. Chen et al. (2017) propose two BFO algorithms for feature subset selection. The algorithms are adaptive chemotaxis BFO and improved swarming elimination-dispersal BFO algorithm. The algorithms are experimented on UCI datasets. The experimental results demonstrate that BFO algorithms are competitive with state-of-the-art algorithms in the literature. Kim, Abraham, and Cho (2007) propose a hybrid BFO algorithm integrated with GA for function optimization problems. The performance of the algorithm is analyzed on mutation, crossover, the lifetime of the bacteria, a variation of step sizes, and chemotactic steps. Hota, Barisal, and Chakrabarti (2010) present a BFO algorithm applied for the solution of the economic and emission load dispatch problem. The BFA is observed to perform in a robust manner in this study. Tang, Wu, and Saunders (2006) develop a BFO algorithm for the optimization of dynamic environments. The results verify that the proposed algorithm gives accurate results in reasonable times.

3.3. Bat algorithm (BA)

BA metaheuristic is first proposed by Yang (2010c). Bats use echolocation (i.e., a type of sonar) to avoid obstacles, detect prey, and locate their nests in the dark. A bat emits a sound and follows the echo that reflects from the objects in the environment. Bats can also detect the difference between food/prey and barriers by using echolocation. The motivation of BA is that this echolocation talent of bats can be formalized as a means to find an optimal solution in an objective function.

BA runs in an iterative fashion. Bats fly with velocity v_i with position x_i having a frequency f_{min}, varying wavelength λ and loudness A_0 while searching for their prey randomly. They can set the frequency of the pulse and adjust its rate $r \in [0, 1]$ (with respect to the proximity of the prey). The loudness is changed from a maximum A_0 value to a minimum value A_{min}. The frequency f in the range of $[f_{min}, f_{max}]$ correlates with a range of wavelengths $[\lambda_{min}, \lambda_{max}]$.

While any wavelength can be used for a run of BA, selection of a suitable wavelength has significant impact on the convergence of the algorithm. In most cases, wavelength is variable during a run, and a range of wavelengths are set up and adjusted accordingly during a run.

The detectable range should be decided that it is comparable to the size of the domain of interest. The frequency can be changed while fixing the wavelength and is related to λ. Parameter f is assumed to be between $[0, f_{max}]$. Higher frequencies have short wavelengths and can travel a shorter distance while lower frequencies have large wavelengths and can travel further. That is, wavelength and frequency relates to both computational cost and exploration capacity of the setting. The rate of pulse is in the range of [0, 1] where 0 means no pulse and 1 means the maximum rate of pulse emission. Algorithm 3 summarizes the execution of a sample BA run.

Algorithm 3. Bat Algorithm (Yang, 2010c)

```
1  Objective function f(x), $x = (x_1, ..., x_d)^T$;
2  Initialize the population of bats $x_i$ ($i$ = 1, 2,..., $n$) and $v_i$;
3  Define pulse frequency $f_i$ at $x_i$;
4  Initialize pulse rates $r_i$ and the loudness $A_i$;
5  while ($t$ < #iterations) do
6      Generate new solutions by setting frequency
7      and updating velocities and locations/solutions [Equations 5-7]
8      if ($rand > r_i$) then
9          Select a solution;
10         Generate a local solution;
11     Generate a new solution by flying randomly;
12     if ($rand < A_i$) & ($f(x_i) < f(x_*)$) then
13         Accept the solution;
14         Increase $r_i$ and reduce $A_i$;
15     Find the best $x_*$;
16 Report the global best result;
```

The positions of virtual bats x_i (i = 1, 2,..., n) and velocities v_i in a d-dimensional space are updated in iterations. New solutions x_i^t and velocities v_i^t at time t are generated with the equations given below.

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{5}$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_*).\, f_i \tag{6}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{7}$$

where $\beta \in [0, 1]$ is a uniform distribution random vector. x_* is the global best location among n bats.

If $\lambda_i f_i$ is the velocity increment, we can use either f_i (or λ_i) to set the new velocity while fixing the factor λ_i (or f_i). $f_{min} = 0$ and $f_{max} = 100$ can be used with respect to the domain of the problem. Each bat is assigned a random frequency initially. It is drawn uniformly from $[f_{min}, f_{min}]$. In the local search, after selecting a solution among the best solutions, a random solution for each bat is generated locally using a random walk process using the formula:

$$x_{new} = x_{old} + \in A^t \tag{8}$$

where $\in [-1, 1]$ is a random value, $A^t = < A_i^t >$ is the average loudness of all the bats at this period of time.

The update of positions and velocities of bats are performed in a similar fashion after each iteration as in a PSO algorithm. Parameter f_i controls the pace and the range of the movement. At each iteration, the loudness A_i and the rate r_i of pulse emission need to be updated. The loudness decreases as bat finds its prey. The rate of pulse emission increases, the loudness can be of any value. The values for A_0 and A_{min} can be assigned as 100 and 1 respectively. For simplicity, $A_0 = 1$ and $A_{min} = 0$ can be applied. $A_{min} = 0$ means that a bat has found the prey and finished emitting a sound.

$$A_i^{t+1} = \alpha A_i^t, \quad r^{t+1} = r_i^0[1 - exp(\gamma t)], \tag{9}$$

where α and $\gamma$are constants.

α is similar to the cooling factor used in the SA (Van Laarhoven & Aarts, 1987). For any 0 <α< 1 and γ> 0, we have

$$A_i^t \to 0, \quad r_i^t \to r_i^0 \quad ast \quad \to \infty \tag{10}$$

Past research suggest that selection $\alpha = \gamma = 0.9$ as parametrization is a suitable assumption during the optimization process (Yang, 2010c). The tuning of the parameters is often done via experimentation, where each bat has different values of loudness and pulse emission rate initially. The first value of loudness A_i^0 can be in the range of [1, 2]. The initial emission rate r_i^0 can be around zero or any value $r_0 \in [0, 1]$. Their loudness and emission rates will be changed only if a new solution is produced.

With many parameters, it is possible to change and control the nature of exploration and exploitation even through a single run, many variations of BA has been proposed in the literature. Gandomi and Yang (2014) propose a chaotic BA to increase the global search capacity of BA for robust optimization. The authors examine different chaotic maps on benchmark problems. The results verify that chaotic BA can outperform the classical versions of BA. Yang and Hossein Gandomi (2012) propose a BA for solving engineering optimization tasks. Eight well-known optimization tasks are carried out and a fair comparison is performed with existing algorithms. Yang (2012a) proposes multi-objective BA for solving design problems. Experimental results verify that the proposed algorithm works efficiently. Yang (2013a) reviews the literature of BA in his survey. The paper gives a review of the BA and its variants. Gandomi, Yang, Alavi, and Talatahari (2013) propose a BA to solve constraint optimization problems. The BA is verified using many benchmark constraint problems. Mirjalili, Mirjalili, and Yang (2014) develop a binary version of BA. A comparative study is carried out with over twenty-two benchmark functions with GA. Khan and Sahai (2012) verify the performance of a BA over other algorithms in neural network training. Fister Jr, Fister, and Yang (2013) present a new algorithm based on BA. The BA is combined with DE strategies. The new algorithm is observed to improve the original BA significantly. Yılmaz and Kücüksille (2015) enhance the search methods of BA through three different means. The results of test sets prove that BA is superior to its standard version. Nakamura et al. (2012) propose a new BA to solve constraint optimization tasks. The performance of BA is verified using several classical benchmark constraint problems.

3.4. Biogeography-based optimization (BBO)

BBO is a population-based metaheuristic proposed by (Simon, 2008). BBO assumes that each individual lives in a habitat with a Habitat Suitability Index (HSI) parameter that measures the fitness of a solution. Two main operators of BBO algorithm are migration and mutation. The migration is a stochastic operator that updates each individual (H_i) by sharing the features of individuals in the population.

The probability of selecting a solution (H_i) as an immigrating habitat is related to its immigration rate λ_i. The probability of selecting the solution H_j as an emigrating habitat is related to the emigration rate, μ_j. Suitability Index Variable (SIV) is a parameter that characterizes the habitat. An SIV is a search parameter and the set of all possible SIVs is the search space. The HSI is decided by the use of SIVs.

Solutions with high fitness values are habitats with a high HSI. Habitats with high HSI have many species, whereas low HSI habitats have fewer species. Each habitat (solution) is governed by an identical species curve and is characterized with an S value which depends on the HSI value of a solution. For example, if there are two habitats characterized by S_1 and S_2, the immigration rate, λ_1 of S_1 will be higher than the immigration rate λ_2 of S_2. The emigration rate, μ_1 of S_1 for will be lower than the emigration rate μ_2 of S_2.

The emigration and immigration rates are used to share information between solutions. Each solution that is based on other solutions is modified with a global probability P_{mod}. If a given solution is selected to be modified, its immigration rate (λ) is used to decide to modify each SIV in that solution. If an SIV is to be modified, the emigration rate of the other solutions is used to select solutions to migrate a random SIV to solution S_i. With an elitist selection, the best solutions are kept from being corrupted by immigration.

Mutation is also used as an exploratory concept in BBO. A habitat's HSI can change due to random events. This is modeled as an SIV mutation whose rates are decided by the species count probabilities that are governed by a differential equation in Eq. (12).

Low and high species counts have low probabilities by Theorem "Medium species counts have higher probabilities since they are close to the equilibrium point". The steady-state value for the probability of the number of each species is;

$$P(\infty) = \frac{v}{\sum_{i=1}^{n+1} v_i} \tag{11}$$

where v and v_i are observation eigenvector.

$$P_s = \begin{cases} -(\lambda_s + \mu_s)P_s + \mu_{s+1}P_{s+1}, & \text{if } S = 0 \\ -(\lambda_s + \mu_s)P_s + \lambda_{s-1}P_{s-1} + \mu_{s+1}P_{s+1}, & \text{if } 1 \leqslant S \leqslant S_{max} - 1 \\ -(\lambda_s + \mu_s)P_s + \lambda_{s-1}P_{s-1}, & \text{if } S = S_{max} \end{cases} \tag{12}$$

Very high and low HSI solutions are equally impossible, whereas solutions with medium HSI are more probable. A mutation rate m that is inversely proportional to the solution probability can be formulated as below:

$$m(S) = m_{max}\left(\frac{1 - P_s}{P_{max}}\right) \tag{13}$$

where m_{max} is a user-defined value. This mutation operator aims to increase diversity. This prevents the local stagnation of the algorithm.

The parameters of a typical BBO are set as follows: $H \in SIV^m$ is a vector of m integers for the solution to an optimization problem. An $SIV \in C$ is a value allowed in a habitat. $C \subset Z^q$ is the set of all integers. $SIV \in C$ and $H \in SIV^m$ are constraints. A habitat suitability index HSI ($H \to R$) is the fitness value of a solution. H^n is a set of habitats and the size of an ecosystem is fixed. The immigration rate $\lambda(HSI)$: $R \to R$ is a monotonically non-increasing function of HSI. λ_i is the rate that SIVs from other habitats migrate into habitat H_i. Emigration rate $\mu(HSI)$: $R \to R$ is a monotonically non-decreasing function of HSI. μ_i is the likelihood of the migration of SIVs from habitat. Habitat modification $\Omega(\lambda, \mu)$: $H^n \to H$ is an operator that tunes habitat based on the ecosystem H^n. The probability of modifying the H is proportional to the immigration rate λ. The source of the modification comes from H_j is proportional to the emigration rate μ_j.

Mutation $M(\lambda, \mu)$: $H \to H$ is a stochastic operator changing habitat SIVs based on the habitat's *a priori* existence probability. An ecosystem transition function $\Psi = (m, n, \lambda, \mu, \Omega, M) = H^n \to H^n$ is a 6-tuple function that changes the ecosystem in iterations. The function can be written as shown in Eq. (14):

$$\Psi = \lambda^n \circ \mu^n \circ \Omega^n \circ HSI^n \circ M^n \circ HSI^n \tag{14}$$

A BBO algorithm is a 3-tuple (BBO = I, Ψ, T) solution to an optimization problem. I: ø $\to H^n$, HSI^n is a function that generates an initial ecosystem of habitats and computes values of each HSI. Ω is the ecosystem transition function and T: $H^n \to \{true, false\}$ is a termination criterion. The pseudocode of the BBO algorithm is presented in Algorithm 4. If $\alpha = 0$ then it is the "*standard BBO algorithm*" (Simon, 2008). If α is random, then it becomes the "*blended BBO*" Algorithm.

Algorithm 4. BBO algorithm (Simon, 2008)

```
1  Create a population, $H_1$, $H_2$,..., $H_n$;
2  Compute HSI values;
3  while (the halting criterion is not satisfied) do
4      Compute the immigration rate $\lambda_i$;
5      for (each habitat (solution)) do
6          for (each SIV (solution feature)) do
7              Select habitat $H_i$ with probability $\propto \lambda_i$;
8              if ($H_i$ is selected) then
9                  Select $H_j$ with probability $\propto \mu_i$;
10                 if ($H_j$ is selected) then
11                     $H_i$ (SIV) $\leftarrow \alpha H_i$ (SIV) + (1- $\alpha$)$H_j$ (SIV);
12             Select $H_i$ (SIV) based on mutation probability $m_i$;
13             if ($H_i$ is selected) then
14                 Replace $H_i$ (SIV) with a randomly generated SIV;
15         Recompute HSI values;
```

A wide range of BBO variations has been proposed in the literature. Bhattacharya and Chattopadhyay (2010) present a hybrid technique combining DE and BBO to solve convex and nonconvex economic load dispatch problems (Gong, Cai, Ling, & Li, 2010). The algorithm aims to improve the quality of the solution and the convergence speed. Ergezer, Simon, and Du (2009) propose a variation to BBO integrated with opposition-based learning. The proposed algorithm outperforms BBO in terms of success rate and the number of fitness evaluation. Ma (2010) studies the generalization of equilibrium species count in biogeography theory. Simon, Ergezer, Du, and Rarick (2011) propose Markov models for BBO with selection, migration, and mutation operators. The models provide theoretically exact limiting probabilities for each possible population distribution. e Silva, Coelho, and Lebensztajn (2012) propose a multiobjective BBO for the constrained design of a wheel motor. The proposed algorithm converges to promising solutions in terms of quality and dominance. A blended migration is proposed for BBO by Ma and Simon (2011). The method is a generalized migration operator for BBO that consists of features from itself and another solution. The blended migration can be observed as an efficient modification when the experimental results are analyzed.

3.5. Cuckoo search algorithm (CSA)

Yang and Deb (2009) propose CSA. The CSA simulates the brood parasitic behavior of cuckoo species with the Lévy flight action of birds and fruit flies. Cuckoo birds have an aggressive breeding attitude. They lay eggs in the nest of other birds and remove the other eggs to increase the hatching chance of their own eggs. In CSA metaheuristic, three simple rules are used. (1) One egg can be laid at a time, and cuckoo leaves its egg in a random nest; (2) Nests having high-quality eggs can survive; (3) The number of host nests is constant, and the egg can be detected by the host bird with a probability $p_a \in [0, 1]$. The egg can be thrown away from nests or the bird can leave the nest, and construct a new nest. The pseudocode of the CSA is presented in Algorithm 5.

Algorithm 5. Cuckoo Search Algorithm (Yang & Deb, 2009)

```
1  Optimization function $f(x)$, $x = (x_1, ..., x_d)^T$
2  Construct an initial population with $n$ nests $x_i$ $(i = 1, 2, ..., n)$;
3  while $(t{+}{+} < stopping\ criterion)$ do
4      Get a random cuckoo by Lévy flights;
5      Calculate its fitness value $F_i$;
6      Select a nest among $n$ (say, $j$) randomly;
7      if $(F_i > F_j)$ then
8          replace $j$ by the new solution;
9      A fraction $(p_a)$ of worse nests are left;
10     New nests are built;
11     Keep the best solutions/nests;
12     Find the current best;
13 Process results;
```

When producing new solutions $x^{(t+1)}$ $(i = 1, 2, \ldots, n)$ for cuckoo bird i, a Lévy flight is realized as in Eq. (15)

$$x_i^{t+1} = x_i^t + \alpha \oplus Lévy(\lambda) \tag{15}$$

where $\alpha < 0$ is the size of a step related to the problem. In most cases, α is selected as 1. The equation above is stochastic to provide a random walk that is a process of Markov Chain with a next location that relies on the current location and the transition probability. The product $\oplus$ means entrywise multiplications. This entrywise product via Lévy flight is more efficient while exploring the problem search space because its length is longer.

The Lévy flight ensures a random walk while the length of the random step is derived from a Lévy distribution with infinite variance and an infinite mean as given in Eq. (16).

$$Lévy \sim u = t^{-\lambda}, \ (1 < \lambda \leqslant 3) \tag{16}$$

A benefit of CSA is that since the number of parameters to be tuned is less than most of the metaheuristic algorithms, it is easily applicable to a wider set of optimization problems. However, in order to ensure that the optimization process does not stick in a local optima, a substantial fraction of the new solutions should be produced by randomization during a run. It is shown in the literature that the randomization of CSA is more efficient as the step length is selected via using a heavy tailed distribution.

Yang (2013b) provides detailed theory and application information in his book about CSA and FA. Shehab, Khader, and Al-Betar (2017) present a comprehensive review of CSA. Its advantages and disadvantages, main architecture, and extended versions are discussed. Gandomi, Yang, and Alavi (2013) propose CSA for solving structural optimization tasks. The algorithm combined with Lévy flights is verified with nonlinear constrained optimization problems.

Yang and Deb (2010) propose a new CSA for some standard test functions and newly designed stochastic test functions. The algorithm is applied to engineering design optimization problems, the design of springs and welded beam structures. Yildiz (2013) proposes CSA for solving manufacturing optimization problems. A milling optimization problem is solved and the results are compared with ACO, AIA, hybrid PSO, and GA.

Ouaarab, Ahiod, and Yang (2014) present an improved discrete CSA for the well-known traveling salesman problem (TSP). The algorithm is tested against a set of benchmarks of symmetric TSP from the well-known TSPLIB library. The results of the tests verify that the algorithm outperforms the other well-known metaheuristics. Walton, Hassan, Morgan, and Brown (2011) propose a modified robust CSA optimization algorithm. Durgun and Yildiz (2012) propose a new CSA for solving structural design optimization problems. This is one of the first applications of the CSA for the shape design optimization problems. Valian, Mohanna, and Tavakoli (2011) propose a strategy for tuning the parameters of CSA. Numerical studies verify that the proposed algorithm can obtain efficient solutions. Basu and Chowdhury (2013) present CSA to optimize convex and nonconvex economic dispatch problems of fossil fuel generators. Tuba, Subotic, and Stanarevic (2011) present a modified CSA for unconstrained optimization problems. The authors develop a modification in which the step size is decided from the sorted, rather than only permuted fitness matrix. Wang, Gandomi, Zhao, and Chu (2016) purpose an enhancement for the search ability of the CSA. The pitch adjustment operation in harmony search is added to the process of the cuckoo updating to speed up convergence. Yang and Deb (2014) review the fundamental ideas of cuckoo search and the latest developments as well as its applications. Bhandari, Singh, Kumar, and Singh (2014) propose CSA for multilevel thresholding using Kapur's entropy. Chandrasekaran and Simon (2012) propose a hybrid CSA with fuzzy system for solving multiobjective unit commitment problem. Three conflicting functions fuel cost, emission and reliability level of the system are considered in the study. The effectiveness of the algorithm is verified with unit test systems. Rajabioun (2011) proposes a CSA for continuous nonlinear optimization problems. The proposed algorithm is applied to benchmark functions and a real problem and it has been proven to be a robust method. Majumder, Laha, and Suganthan (2018) propose a hybrid CSA for the solution of the scheduling identical parallel batch processing machines. The authors claim that the makespan for the scheduling problem is minimized.

3.6. *Firefly algorithm (FA)*

The FA is proposed by Yang (2010a). FA is inspired by the behavior of the short and rhythmic flashing characteristics of fireflies. Two main functions of such flashes attract mating partners or warning against predators. The rhythmic flash, the rate of flashing brings sexes together. The flashing can be formulated as a function to be optimized for combinatorial algorithms. These flashing characteristics are idealized by the following rules. (1) All fireflies can attract other fireflies without any concern about their gender. (2) Attractiveness is the brightness of the firefly. Therefore, the less brighter firefly moves towards brighter ones. They decrease the attractiveness as the distance between fireflies increases. It moves randomly when there is no brighter one. (3) The search space of the objective function affects the brightness of a firefly. Algorithm 6 depicts a typical run of FA.

Two crucial issues of FA are the light intensity and formulation of the attractiveness. The brightness of the firefly decides its attractiveness where it is the encoded objective function. The brightness of a firefly (I) at a location x can be chosen as $I(x) \propto f(x)$ and the attractiveness β is relative, it will be decided by other fireflies in the population. The brightness varies with respect to the distance r_{ij} between two fireflies. The density of the light intensity $I(r)$ changes according to the inverse square law given below:

$$I(r) = \frac{I_s}{r^2} \tag{17}$$

where I_s is the intensity at the source, γ, is a coefficient of fixed light absorption. The light intensity I changes with the distance r. I_0 is the original light intensity.

$$I = I_0 e^{-\gamma r} \tag{18}$$

The effect of absorption and the inverse square law is approximated as a Gaussian form:

$$I(r) = I_0 e^{-\gamma r^2} \tag{19}$$

The attractiveness (β) of a firefly is defined by:

$$\beta = \beta_0 e^{-\gamma r^2} \tag{20}$$

where (β_0) is the attractiveness at $r = 0$. The above function can be approximated as:

$$\beta = \frac{\beta_0}{1 + \gamma r^2} \tag{21}$$

Eqs. (20) and (21) have a characteristic distance $\Gamma = 1/\sqrt{\gamma}$ where the attractiveness changes from β_0 to $\beta_0 e^{-1}$ for Eq. (20) or $\beta_0/2$ for Eq. (21) significantly.

The attractiveness $\beta(r)$ is a function that monotonically decreases:

$$\beta(r) = \beta_0 e^{\gamma r^m}, (m \geqslant 1). \tag{22}$$

For a constant γ, the characteristic length is:

$$\Gamma = \gamma^{-1/m} \to 1, m \to \infty \tag{23}$$

For a given length scale Γ, the parameter γ can be used as an initial value:

$$\gamma = \frac{1}{\Gamma^m} \tag{24}$$

The distance between the locations of fireflies i and j ($i = 1, 2, \ldots, n$) is the Cartesian distance (x_i and x_j are the locations of the fireflies):

$$r_{ij} = \left\| x_i - x_j \right\| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \tag{25}$$

where $x_{i,k}$ is the k^{th} element of the coordinate x_i of firefly i.

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{26}$$

The movement of firefly i is attracted to a brighter firefly j is:

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha \epsilon_i \tag{27}$$

The third term is a random α parameter, and ϵ_i is a random vector of values derived from a Gaussian or uniform distribution. The simplest form, ϵ_i, can be swapped with $rand - 1/2$ where $rand$ is a number generator distributed in [0, 1] uniformly. $\beta_0 = 1$ and $\alpha \in [0, 1]$ are assumed during the optimization process. Eq. (27) is a random walk towards brighter fireflies. When β_0 is selected as 0, it becomes a random search. The randomization term can be implemented with other distribution methods such as Lévy flights. The distributional properties of randomization is an important facility for controlling the balance between exploration and exploitation, and attracted a lot of attention in the literature (Fister, Fister, Yang, & Brest, 2013; Gandomi, Yang, Talatahari, & Alavi, 2013).

The parameter γ decides the diversity of the attractiveness. Its value affects the convergence of the FA. In theory, the value of $\gamma \in [0, \infty)$, but in an application, $\gamma = O(1)$ is used by the characteristic length Γ of the function to be optimized. The value changes from 0.1 to 10.

Algorithm 6. Firefly Algorithm (Yang, 2010a)

```
1  f(x) is an objective function where, $x = (x_1, ..., x_d)^T$
2  // $n$ is the number of fireflies
3  Generate a population of fireflies $x_i$ ($i$ = 1, 2, ..., $n$);
4  Define coefficient $\gamma$ of light absorption;
5  while ($t <$ Max Generation) do
6      for ($i <$ all $n$ fireflies) do
7          for ($j < n$ fireflies) do
8              Light intensity $I_i$ at $x_i$ is determined by f($x_i$);
9              if ($I_j > I_i$) then
10                 Move firefly $i$ towards $j$ in all $d$ dimensions;
11             Attractiveness varies with distance $\gamma$ via exp[-$\gamma r$];
12             Calculate new solutions and update intensity of light;
13     Report the current best solution;
```

There exist numerous scientific FA studies and adaptations in the literature. Yang et al. (2013) review the basics of FA in their study comprehensively. The authors discuss the importance of balancing exploration and exploitation. Gandomi, Yang, Talatahari, et al. (2013) introduce chaos into FA to improve its global search for robust optimization. Chaotic maps are implemented to set the attractive motion of fireflies. Wang, Wang, et al. (2017) present a new FA for the well-known benchmark functions. The results verify that the proposed algorithm improves the accuracy of solutions and reduce the execution time. Gandomi, Yang, and Alavi (2011) developed a novel FA for solving continuous/discrete structural optimization problems. The results confirm the validity of the algorithm. Senthilnath, Omkar, and Mani (2011) develop a new FA for clustering problems. The performance of the FA is compared with ABC, PSO, and other important methods in literature. Yang, Hosseini, and Gandomi (2012) propose a new FA for economic dispatch problems. Yang (2013) enhances the FA to solve multiobjective optimization problems. The author validates the new algorithm using a selected subset of test functions. Farahani, Abshouri, Nasiri, and Meybodi (2011) propose FA to stabilize the movement of a firefly. A new behavior to direct fireflies to global best is recommended if there is no any better solution in the environment. Jati et al. (2011) introduce a new FA for the solution of traveling salesman problem (TSP). Fister Jr, Yang, Fister, and Brest (2012) propose a hybrid FA with a local search technique for the well-known combinatorial optimization problems. The results of the proposed algorithm are very promising and have great potential to be applied to other combinatorial optimization problems successfully. Zubair and Mansor (2019) propose a FA for the optimization of computer-aided process planning turning machining parameters for cutting tool selection.

3.7. Gravitational search algorithm (GSA)

Rashedi et al. (2009) propose GSA. In this metaheuristic, search agents are objects and their success is proportional to their masses. The objects pull one another by the force of gravity. This force causes the movement of light agents toward heavier mass agents. The communication of agents is provided through gravitational force. The exploitation for the GSA is guaranteed by heavy masses that move slowly. Each object has a position, an inertial mass, a passive and an active gravitational mass. Each object represents a solution that is directed by setting the gravitational and inertia masses. The heaviest agent is the current best solution and other agents are attracted by this agent. GSA applies the Newtonian laws of gravitation and motion. Each object attracts every other one and the gravitational force between two objects is proportional to the product of their masses and inversely proportional to the distance between them, R. In order to be computationally effective, GSA uses the value R instead of R^2. The law of motion is that the current velocity is equal to the total sum of the fraction of its previous velocity and the change in the velocity. I.e., in an environment with N objects, the position of object i is:

$$X_i = X_i^1, \; ..., X_i^d, \; ..., X_i^n, \quad for \quad i = 1, 2, \; ..., N, \tag{28}$$

where x_i^d is the position of object i in the d^{th} dimension. The force applied to object 'i' from agent 'j' at time t is:

$$F_{ij}^d(t) = G(t)\frac{M_{pi}(t) * M_{aj}(t)}{R_{ij}(t) + \varepsilon}\left(X_j^d(t) - X_i^d(t)\right) \tag{29}$$

where M_{aj} is the gravitational mass applied to agent j, M_{pi} is the passive gravitational mass applied to agent i, $G(t)$ is gravitational at time t, ε is a small constant, and $R_{ij}(t)$ is the Euclidian distance between objects i (i = 1, 2, …, N) and j (j = 1, 2, …, N):

$$R_{ij}(t) = \|X_i(t), X_j(t)\|_2 \tag{30}$$

The total force that is applied to object i in d is a random sum of d^{th} components of the forces from other objects:

$$F_i^d(t) = \sum_{j=1, j\neq i}^{N} rand_j F_{ij}^d(t) \tag{31}$$

where $rand_j$ is a number in [0, 1]. The acceleration of the object i at time t, and in direction d^{th}, $a_i^d(t)$ is given as:

$$a_i^d(t) = \frac{F_{ij}^d(t)}{M_{ii}^d(t)} \tag{32}$$

where M_{ii} is the inertial mass of object i. The new velocity of an object is a fraction of its current velocity and its acceleration. Its position and velocity are calculated as follows:

$$v_i^d(t+1) = rand_i * v_i^d(t) + a_i^d(t) \tag{33}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{34}$$

where $rand_i$ is a uniform variable in [0, 1]. The constant, G, is initialized and reduced with time to control the accuracy of the search.

$$G(t) = G(G_0, t) \tag{35}$$

A heavier mass indicates an efficient object (agent). Better solutions are represented as heavier objects that have higher attractions and move more slowly. The gravitational and inertial masses are updated by the equations given below:

$$M_{ai} = M_{pi} = M_{ii} = M_i; i = 1, 2, \; ..., N; \tag{36}$$

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \tag{37}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{38}$$

where $fit_i(t)$ represent the fitness value of the agent i at time t, and, *worst(t)* and *best(t)* are defined as follows (for a minimization problem):

$$best(t) = \min_{j \in 1,\ldots,N} fit_j(t) \tag{39}$$

$$worst(t) = \max_{j \in 1,\ldots,N} fit_j(t) \tag{40}$$

It is to be noted that for a maximization problem, Eqs. (39) and (40) are changed to Eqs. (41) and (42) respectively:

$$best(t) = \max_{j \in 1,\ldots,N} fit_j(t) \tag{41}$$

$$worst(t) = \min_{j \in 1,\ldots,N} fit_j(t) \tag{42}$$

In order to provide a balance between exploration and exploitation, the number of agents with a lapse of time in Eq. (31) should be reduced. To avoid getting into local optima the GAS uses the exploration at initial phases. The level of exploration should be decreased and exploitation should be increased throughout the iterations. In order to improve the efficiency of GSA, the K_{best} agents should attract the others and thus, K_{best} is a function that changes with time. It is initially set to K_0 at the beginning and modified in a monotonically decreasing fashion. The Eq. (31) is formalized as:

$$F_i^d(t) = \sum_{j \in K_{best}, j \neq i}^{N} rand_j F_{ij}^d(t) \tag{43}$$

where K_{best} is the set of heaviest *K* objects with the best fitness value and the largest mass. The pseudocode of GSA is given in the Algorithm 7.

Algorithm 7. Gravitational Search Algorithm (Rashedi et al., 2009)

```
1 Generate initial population
2 while (t < stopping criterion) do
3     Calculate the fitness of all search agents
4     Update G(i), best(i), worst(i) for i = 1,2,. . .,N.
5     Calculation of acceleration and $M_i(t)$ or each agent $i$
6     Update velocity and position
7     t=t+1
8 Return the best solution
```

Some of the recent studies related to GSA are listed as follows: Rashedi, Nezamabadi-Pour, and Saryazdi (2010) present a binary GSA. Rashedi, Nezamabadi-Pour, and Saryazdi (2011) examine the presentation of a new linear and nonlinear filter modeling based on a GSA. Duman, Güvenç, Sönmez, and Yörükeren (2012) propose a GSA to find the optimal solution for optimal power flow of a power system. Mirjalili, Hashim, and Sardroudi (2012) propose a hybrid GSA as a new training method for Feedforward Neural Networks to search the performance of algorithms to prevent sticking in local optima and the slow convergence of evolutionary learning algorithms. Li and Zhou (2011) propose a new GSA optimization algorithm for the parameters identification of hydraulic turbine governing system. Hatamlou, Abdullah, and Nezamabadi-Pour (2012) present a hybrid GSA for data clustering problems. Hassanzadeh and Rouhani (2010) propose a new multi-objective GSA for different test benches. The results prove the superiority of the algorithm. Sabri, Puteh, and Mahmood (2013) give a review to provide an outlook on GSA. Rashedi, Rashedi, and Nezamabadi-pour (2018) give a recent comprehensive survey on GSA.

3.8. Grey wolf algorithm (GWO)

Mirjalili, Mirjalili, and Lewis (2014) propose GWO metaheuristic in 2014. The Grey Wolf is a member of predator animals family and lives with a pack. Each wolf pack has a social hierarchy. A typical wolf hierarchy contains several types of wolves such as the "*alpha dog*", "*beta dog*", "*omega dog*", and "*subordinates*". The alpha dog in the pack has the one with the most responsibilities. It is dominant and leads the pack. Beta is the second level dog in the hierarchy. He/she is the most likely one to be the alpha dog in case alpha dog becomes dysfunctional. Omega dogs are the lowest ranking ones. A dog is called subordinate (or delta) if it is not one of the dogs mentioned above. Group hunting is the most interesting swarm behavior of these wolves.

In the mathematical modeling of the GWO, the social hierarchy, tracking, encircling, and attacking prey are the key points of the GWO algorithm. In this model, the best solution is considered to be alpha dog. The second and the third best solutions are beta and delta dogs respectively. The rest of the swarm is called omega dogs. For the phase of encircling the prey, grey wolves encircle the prey. To model the encircling the prey, the Eqs. (44) and (45) are used.

$$\overrightarrow{D} = |\overrightarrow{C} \cdot \overrightarrow{X_p}(t) - \overrightarrow{X}(t)| \tag{44}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X_p}(t) - \overrightarrow{A} \cdot (\overrightarrow{D}) \tag{45}$$

t is the iteration index, $\overrightarrow{A}$ and $\overrightarrow{C}$ are coefficient vectors, $\overrightarrow{X_p}$ is the prey position, and $\overrightarrow{X}$ is the grey wolf position. $\overrightarrow{A}$ and $\overrightarrow{C}$ are given as below:

$$\overrightarrow{A} = 2\overrightarrow{a} \cdot r_1 - \overrightarrow{a} \tag{46}$$

$$\overrightarrow{C} = 2 \cdot r_2 \tag{47}$$

r_1 and r_2 are vectors in the range of [0,1]. During the iterations, components of $\overrightarrow{a}$ are decreased from 2 to 0. In this concept, grey wolves move around the best solution in hyper-cubes (i.e., candidate solutions each varying only in few dimensions from the best solution) within an *n* dimensional space. Grey wolves can detect the location of prey and encircle it. The alpha dog leads the hunt. Other dogs also take part in hunting. In the mathematical simulation of hunting, the best three solutions update the position of other search agents. The equations below are given for the process.

$$\overrightarrow{D_\alpha} = |\overrightarrow{C_1} \cdot \overrightarrow{X_\alpha} - \overrightarrow{X}|,\ \overrightarrow{D_\beta} = |\overrightarrow{C_2} \cdot \overrightarrow{X_\beta} - \overrightarrow{X}|,\ \overrightarrow{D_\delta} = |\overrightarrow{C_3} \cdot \overrightarrow{X_\delta} - \overrightarrow{X}| \tag{48}$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1} \cdot (\overrightarrow{D_\alpha}),\ \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2} \cdot (\overrightarrow{D_\beta}),\ \overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3} \cdot (\overrightarrow{D_\delta}) \tag{49}$$

$$\overrightarrow{X}\left(t+1\right) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \tag{50}$$

For the attacking phase (exploitation) of the algorithm, the value of $\overrightarrow{a}$ is decreased gradually. So the fluctuation rate of $\overrightarrow{A}$ decreases by $\overrightarrow{a}$, since every local search algorithm is prone to local stagnation. Therefore, GWO uses an efficient exploration method: Alpha, beta, and delta dogs always try to stay far away from each other. This provides a good diversity in the problem search space.

For $\overrightarrow{A}$, values between 1 and −1 are used. This provides a global exploration capability for the GWO algorithm. For $\overrightarrow{C}$, random values between [0, 2] are used. Local optima avoidance and efficient exploration are provided in this way. After generating a random population, alpha, beta, and delta dogs calculate the position of the best prey. In order to choose exploration and exploitation, parameter *a* is decreased from 2 to 0 respectively. The GWO algorithm terminates when the criterion is satisfied. The pseudocode of the GWO algorithm is presented in Algorithm 8.

Algorithm 8. Grey Wolf Algorithm (Mirjalili, Mirjalili, & Lewis, 2014)

```
1  Produce initial Grey Wolf population $X_i$ ($i$=1,2,...,$n$);
2  Give initial values to $a$, $A$, and $C$ randomly;
3  Computer the fitness values of agents in the population;
4  $X_\alpha$ = the best agent;
5  $X_\beta$ = the second best agent;
6  $X_\delta$= the third best agent;
7  while ($t < \#iterations$) do
8      for (each agent) do
9          Update the position of the current search agent (Equation 50);
10     Update $a$, $A$, and $C$;
11     Update the fitness value of agents;
12     Update $X_\alpha$ , $X_\beta$ , and $X_\delta$;
13     t++;
14 Return $X_\alpha$;
```

Some of the recent studies related to GWO are as follows: Mirjalili, Saremi, Mirjalili, and Coelho (2016) propose a Multiobjective GWO to optimize problems with multiple objectives. An external archive is combined with GWO to save and retrieve Pareto optimal results. Komaki and Kayvanfar (2015) propose a GWO algorithm for the assembly flow shop scheduling problem with a release time of jobs that can be applied to industrial engineering problems easily. Emary, Zawbaa, and Hassanien (2016) propose a novel binary version of GWO for feature subset selection of data classification problems. Mittal, Singh, and Sohi (2016) propose a modified GWO to balance the exploration and exploitation efforts of GWO that improves the performance of the algorithm. Kohli and Arora (2018) introduce the chaotic GWO algorithm to accelerate its global convergence speed. Experiments are studied to carry out to solve standard constrained benchmark problems. Song et al. (2015) propose a novel GWO for surface wave dispersion curve inversion scheme. The proposed algorithm is tested on noise-free, noisy, and field data. For verification, the results are compared to GA, PSO, and GSA. The algorithm is reported to be efficient. Qin et al. (2019) propose a hybrid discrete GWO for the casting production scheduling problem with multiobjective and multi-constraint.

3.9. Harmony search (HS)

HS is a metaheuristic algorithm based on musical compositions and the process of writing a composition (Geem et al., 2001). HS is proposed by Yang (2009) in 2001 and has been applied to numerous optimization problems since then (Manjarres et al., 2013). HS makes use of methods applied by musicians to create harmonic musical compositions in order to model optimization problems (Wang, Gao, & Zenger, 2015). In HS, a musician has three possible choices when improvising a song: (1) playing any well-known piece of music (pitches in harmony) naturally from his or her memory; (2) playing music similar to an existing piece (by adjusting the pitch); or (3) composing random harmonic notes. Geem et al. (2001) use these possible choices during the optimization process of a problem, applying, pitch adjusting, and randomization.

The usage of harmony memory is similar to the selection of best chromosomes in GA. The harmony memory ensures to keep the best harmonies to new harmony memory. It is assigned as a parameter, $r_{accept} \in [0, 1]$ that is called harmony memory accepting rate. When the rate is too small, a few best harmonies are selected and this causes slower convergence of the HS algorithm. When the rate is too high (a value close to 1), it may not be possible to explore all the harmonies well. This can lead to wrong solutions. The parameter r_{accept} is selected between [0.7, 0.95] to prevent his problem.

The pitch adjustment is the second parameter determined by the bandwidth (b_{range}) and the adjusting rate of a pitch r_{pa}. Pitch adjustment changes the frequencies and generates diversity in the HS. Linear or nonlinear adjustment is used to set the pitch value.

$$x_{new} = x_{old} + b_{range} * \varepsilon \tag{51}$$

x_{old} is the current pitch, and x_{new} is the new solution after the adjustment of pitch. This process generates a neighboring solution to the existing solution by changing the pitch slightly. Pitch adjustment mimics like the mutation operator in evolutionary algorithms. A parameter (pitch-adjusting rate r_{pa}) can be used to control the adjustment level. A small adjustment rate can slow the convergence time of HS, whereas a high adjustment rate can act as a random search process. A value between [0.1, 0.5] is observed to be a good balance for r_{pa}.

The third parameter (randomization) is used to provide diversified solutions. The randomization enables the system to explore different solutions. The randomization can direct the search to explore various different solutions to obtain the global optimal solutions. the probability of randomization is given below:

$$p_{random} = 1 - r_{accept} \tag{52}$$

where the probability of adjusting pitches are:

$$p_{pitch} = r_{accept} * r_{pa} \tag{53}$$

Algorithm 9 summarizes a typical HS.

Algorithm 9. Harmony Search Optimization Algorithm (Geem et al., 2001).

```
1 Generate initial harmonics;
2 Introduce pitch adjusting rate ($r_{pa}$), pitch limits and bandwidth;
3 Introduce harmony memory accepting rate ($r_{accept}$);
4 while ($t <$ #iterations) do
5     Generate harmonics by accepting the best harmonics;
6     Tune pitch to get new harmonics/solutions;
7     if (rand$> r_{accept}$) then
8         choose a random harmonic from population;
9     else if (rand$> r_{pa}$) then
10        tune the pitch within limits randomly;
11    else
12        generate new harmonics randomly;
13 Find the current best harmonics;
```

Some of the well-known studies of HS are as follows: Lee and Geem (2005) describe an HS algorithm for engineering optimization problems having real-number design variables. The proposed algorithm searches the space with a perfect state of harmony. It uses a random search instead of a gradient process so that derivative information becomes useless. Wang and Huang (2010) examine the main difficulties of HS while selecting suitable parameters. They use consciousness to tune the parameters. The classical number generator is updated with the low-discrepancy sequences for initial harmony memory. Omran and Mahdavi (2008) propose a global-best HS (GHS) to improve the performance. GHS algorithm outperforms other algorithms when applied to benchmark problems. Al-Betar, Doush, Khader, and Awadallah (2012) propose novel selection schemes. Zou, Gao, Wu, and Li (2010) use a novel global HS (NGHS) algorithm to optimize unconstrained problems. The NGHS algorithm includes position updating and genetic mutation. Shabani, Mirroshandel, and Asheri (2017) propose an algorithm that makes use of experienced musicians. When making harmony, the musicians modify the undesired notes of the current harmony instead of throwing them away. This method is used to allow the HS to exploit the information obtained in the harmony memory to improve current harmonies. The algorithm is Selective Refining HS in which a new harmony memory is utilized. Kumar, Chhabra, and Kumar (2014) present a parameter adaptive HS for solving optimization problems. The two important parameters of HS are changed dynamically in the proposed algorithm. Lee and Geem (2004) describe a new HS that does not need any initial values and applied a random search instead of a gradient process. Geem (2006) presents a cost minimization algorithm for the water distribution networks. The model uses an HS algorithm to satisfy the constraints. Lee, Geem, Lee, and Bae (2005) propose an efficient optimization algorithm for structures with discrete variables HS. Alatas (2010) proposes HS algorithms with chaotic maps to tune parameters and improve the convergence properties to prevent the HS to get stuck into local optima. The studies above verify that HS and its variant algorithms are global search algorithms that can be applied to engineering optimization problems. Exploration and exploitation strategies of HS are different from classical metaheuristic algorithms. Although HS is not a parametrically sensitive algorithm, studies reveal that tuning the parameters of HS improves its performance. HS is reported to be a robust optimization algorithm for solving NP-Hard engineering optimization problems.

3.10. Krill herd (KH)

Gandomi and Alavi (2012) propose KH metaheuristic. One of the most important features of KH is that they can construct large groups (Hardy, 1935). When other sea animals attack a herd, they can eat individual krills but this only reduces the density of the herd. The purpose of a KH is multiobjective that it tries to increase the density of the herd while reaching the food. Each krill moves toward the best global solution while searching for the highest density and the food. During the attacks to the KH, individuals are removed from the swarm. This process decreases the average density of the KH and the distance of the krill swarm from the location of the food.

The fitness of an individual is evaluated with the distance of the KH from the food and the density of the group. The location of an individual krill in 2D is decided by the movements of other krill individuals' foraging for food, and random diffusion. A Lagrangian model is used to be able to search the whole space with n dimensions (Eq. (54)):

$$\frac{dX_i}{dt} = N_i + F_i + Di \tag{54}$$

where N_i is the action started by the other individuals; F_i is the act of foraging, and D_i is the physical diffusion of the krill i ($i = 1, 2, \dots, n$). Individuals in the herd try to keep a high density and act with the effects of other members of the herd. The parameter, direction of motion, α_i, is effected by the density of local swarm, a target swarm, and a repulsive swarm. Individual movement of each krill can be defined:

$$N_i^{new} = N^{max}\alpha_i + \omega_n N_i^{old} \tag{55}$$

where

$$\alpha_i = \alpha_i^{local} + \alpha_i^{target} \tag{56}$$

$N^{max}$is the maximum speed, α_n is the inertia weight of the motion in the range $[0, 1]$, $\omega_n N_i^{old}$ is the last motion induced, α_i^{local} is the local effect of neighbors and α_i^{target} is the target direction effect by the best krill individual. The effect of the neighbors can be an attractive/repulsive

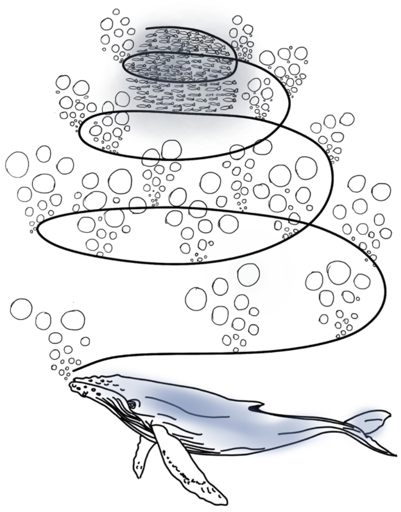
Fig. 5. The bubble-net feeding behavior of a humpback whale.

likelihood between the individuals during a local search. Such behavior utilizes for controlling the balance between exploration and exploitation in KH: allowing more attraction between individuals improves exploitation while allowing repulsive behavior within the herd allows a more diverse search space, thus exploration.

The foraging activity of the algorithm is formulated with two parameters, the food location, and the past information about the food location. This motion of krill i can be given as in Eq. (57).

$$F_i = V_f \beta_i + \omega_f F_i^{old} \tag{57}$$

where

$$\beta_i = \beta_i^{food} + \beta_i^{best} \tag{58}$$

V_f is the velocity of foraging, ω_f is the inertia weight of the foraging movement in the range $[0, 1]$, β_i^{food} is the food attractive and β_i^{best} is the impact of the best fitness of krill i. *The physical diffusion* is a random procedure. This can be explained in terms of a maximum diffusion speed and a random directional vector. The formulation can be given as below:

$$D_i = D^{max}\delta \tag{59}$$

where D^{max} is the maximum diffusion speed, and δ is the random vector and the random values are in the $[-1, 1]$.

The motions direct the location of a krill toward the krill with the best fitness value. According to the formulations for krill i, if the fitness value of each effective factors (K_j, K^{best}, K^{food} or K_i^{best}) is better than the fitness of krill i, it has an attractive effect. It is obvious from the formulations that a better fitness is more effective on the movement of krill i individual. The physical diffusion carries out a random walk in the method. Using different parameters of the motion, the position vector of a krill during the interval t to $t + \Delta t$is:

$$X_i\left(t + \Delta t\right) = X_i(t) + \Delta t\frac{dX_i}{dt} \tag{60}$$

KH algorithm also uses operators of crossover and mutation. An adaptive crossover operator is employed in the KH algorithm. The mutation is controlled by a probability. The pseudocode of the KH algorithm is presented in Algorithm 10.

Algorithm 10. Krill Herd Algorithm (Gandomi & Alavi, 2012)

```
1 Describe the simple bounds, determine the parameters;
2 Create the initial population randomly
3 Evaluate the fitness value of each krill;
4 while (Stopping criterion is not satisfied) do
5     Motion effected by the krill;
6     Foraging motion;
7     Physical diffusion;
8     Use crossover and mutation operators;
9     Update the krill individual position;
```

Some of the recent studies on KH algorithm are presented as follows: Wang, Guo, Gandomi, Hao, and Wang (2014) introduce the chaos theory for KH optimization process to accelerate its convergence speed. Different chaotic maps are used in the proposed chaotic method to set the movements of the krill. Wang, Gandomi, and Alavi (2014) present Stud KH (SKH) optimization method to global optimization. An updated genetic reproduction operator is introduced into the KH during the krill updating process. Wang, Guo, et al. (2014) propose a novel hybrid KH algorithm to solve global numerical optimization problem. The algorithm integrates the exploration of harmony search (HS) with the exploitation of KH. Wang, Gandomi, Alavi, and Hao (2014) deal with the poor exploitation characteristic of the KH algorithm. They propose a hybrid DE algorithm. Wang et al. (2013) develop a Lévy-flight KH algorithm for solving optimization problems within reasonable execution times. The combination of a new local Lévy-flight operator for the process improves the efficiency with global numerical optimization problems. Wang, Hossein Gandomi, and Hossein Alavi (2013) improve the performance of KH algorithm. A series of chaotic PSO-KH algorithms are proposed for solving optimization problems. Guo, Wang, Gandomi, Alavi, and Duan (2014) present an improved KH algorithm to solve global optimization problems. The method exchanges information between the best performing krill during the motion calculation process. Detailed information about KH can be found in a recent survey by Bolaji, Al-Betar, Awadallah, Khader, and Abualigah (2016).

3.11. Social spider optimization (SSO)

SSO is a metaheuristic algorithm proposed by Cuevas et al. (2013). Even though most of the spiders are solitary, the members of social-spider species demonstrate and may show cooperative behavior. The social-spiders have a tendency to live in groups and each member in a group has a variety of tasks such as mating, hunting, web design, and social interaction. The web is a crucial part of the colony and it is used as a communication means. A web is employed by each spider to manage its own cooperative behavior. The SSO algorithm is inspired by the cooperative characteristics of social-spiders. The interaction of individual spiders (solutions) are simulated depending on the biological laws of a cooperative spider colony. Agents are considered as male and female by the SSO algorithm. Such an approach allows not only to simulate the cooperative behavior of the colony in a better way but also to prevent critical problems faced in the classical metaheuristics. These are the incorrect exploration-exploitation balance and premature convergence. The search space is assumed to be a communal web by the SSO algorithm. In this communal web, all the social-spiders interact with each other. A spider's position is considered to be a solution and every spider has a fitness value (weight) of the solution.

The colony of the social-spiders is a highly female-biased population. The number of females N_f is randomly selected within the range of 65–90% of the entire population N. Therefore, N_f is evaluated as:

$$N_f = floor\left[(0.9 - rand\cdot 0.25)\cdot N\right] \tag{61}$$

where rand is a random number between $[0, 1]$ and the number of males N_m is considered as $N_m = N - N_f$.

Each individual (spider) receives a weight w_i that represents the solution quality of the spider i in population $\mathbf{S}$. The weight of every spider is calculated as follows:

$$w_i = \frac{J\left(\mathbf{s}_i\right) - worst_{\mathbf{s}}}{best_{\mathbf{s}} - worst_{\mathbf{s}}} \tag{62}$$

where $J\left(\mathbf{s}_i\right)$ is the fitness value of the spider position $\mathbf{s}_i$ with regard to the objective function J(·).

Information exchange is managed by a communal web mechanism. This is important for collective coordination of the population and encoded as vibrations that depend on the weight and distance of the spider which generates them. Vibrations perceived by the individual i $(i = 1, 2, \dots, n)$ as a result of the information transmitted by the member j has been modeled according to the following equation;

$$Vib_{i,j} = w_j\cdot e^{-d_{i,j}^2} \tag{63}$$

where the $d_{i,j}$ is the Euclidean distance between the spiders i *and* j, such that $d_{i,j} = \|\mathbf{s}_i - \mathbf{s}_j\|$

Three special relationships are considered within the SSO approach;

- Vibrations $Vibc_i$ are perceived by the individual **i** ($\mathbf{s}_i$) as a result of the information transmitted by the member $c(\mathbf{s}_c)$ (the nearest member to i and possesses a higher weight in comparison to i).
- Vibrations $Vibb_i$ are perceived by the individual **i** ($\mathbf{s}_i$) as a result of the information transmitted by the member $b(\mathbf{s}_b)$ (the individual

with the best weight (best fitness value) of the entire population **S**).

- Vibrations $Vibf_i$ are perceived by the individual **i** ($\mathbf{s}_i$) as a result of the information transmitted by the member $f(\mathbf{s}_f)$ (being the nearest female individual to i).

The SSO initializes the entire population including random female and male members. Each spider position, f_i or m_i, is a n-dimensional vector containing the parameter values to be optimized.

Female cooperative operator: Social-spiders perform cooperative interaction with other members. Female spiders present an attraction or dislike. Emulation of the cooperative behavior of the female spider is performed by an operator which considers the position change of the female spider i at each iteration. Any position change can be of *attraction or repulsion*. These can be a combination of three different elements. The first one involves the change regarding the nearest member to i that holds a higher weight and produces the vibration $Vibc_i$. The second one considers the change regarding the best individual of the entire population **S** who produces the vibration $Vibb_i$. Finally, the third one includes a random movement.

For random movement, either attraction or repulsion, a uniform random number r_m is generated within the range [0, 1]. If r_m is smaller than a threshold PF, an *attraction* movement is generated; otherwise, a *repulsion* movement is produced. Therefore, such operator can be modelled as follows:

$$\mathbf{f}_i^{k+1} = \begin{cases} \mathbf{f}_i^k + \alpha\cdot Vibc_i\cdot(\mathbf{s}_c - \mathbf{f}_i^k) + \beta\cdot Vibb_i\cdot(\mathbf{s}_b - \mathbf{f}_i^k) \\ \quad + \delta\cdot(rand - \frac{1}{2}),\ with\ probability\ PF \\ \mathbf{f}_i^k + \alpha\cdot Vibc_i\cdot(\mathbf{s}_c - \mathbf{f}_i^k) + \beta\cdot Vibb_i\cdot(\mathbf{s}_b - \mathbf{f}_i^k) \\ \quad + \delta\cdot(rand - \frac{1}{2}),\ with\ probability\ (1-PF) \end{cases} \tag{64}$$

where α, β and δ and rand are random numbers in the range of [0, 1] and k represents the iteration number. The individual s_c and s_c represent the nearest member to i that holds a higher weight and the best individual of the entire population **S**, respectively.

Male cooperative operator: Male members in SSO are divided into two different groups which are *dominant members* **D** and non-dominant members **ND** according to their position with regard to the median member. Male members, with a weight value above the median value within the male population, are considered the dominant individuals **D**. Those under the median value are depicted as non-dominant **ND** males.

In order to implement such computation, the male population **M** ($\mathbf{M} = \mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_{N_m}$) is arranged according to their weight values in decreasing order. Thus, the individual whose weight $w_{N_{f_m}}$ is located in the middle is considered the median male member. Since indexes of the male population **M** in regard to the entire population **S** are increased by the number of female members N_f, the median weight is indexed by N_{f_m}. With respect to the computation, the male spider position can be modelled as follows:

$$\mathbf{m}_i^{k+1} = \begin{cases} \mathbf{m}_i^k + \alpha\cdot Vibf_i\cdot(s_f - \mathbf{m}_i^k) + \delta\cdot(rand - \frac{1}{2}),\ w_{N_f+i} > w_{N_f+m} \\ \mathbf{m}_i^k + \alpha\cdot\left(\dfrac{\sum_{h=1}^{N^m} \mathbf{m}_h^k\cdot w_{N_f+h}}{\sum_{h=1}^{N^m} w_{N_f+h}} - \mathbf{m}_i^k\right),\ w_{N_f+i} \leqslant w_{N_f+m} \end{cases} \tag{65}$$

Mating operator: Mating in a social-spider colony is performed by dominant males and the female members. When a dominant male $\mathbf{m}_g$ spiders ($g \in \mathbf{D}$) locates a set $\mathbf{E}^g$ of female members within a range of r (range of mating), it mates, generates a new spider s_{new} which is generated considering all the elements of the set $\mathbf{T}_g$ that, in turn, has been generated by the union $\mathbf{E}^g \cup \mathbf{m}_g$. In the mating process, the weight of each involved spider (elements of $\mathbf{T}_g$) defines the probability of influence for each individual into the new brood. The spiders holding a heavier weight are more likely to influence the new product, while elements with lighter weight have a lower probability. Details of the operators and equations of SSO can be found in a study by Cuevas et al. (2013). The pseudocode of SSO algorithm is presented in Algorithm 11.

Algorithm 11. Social spider optimization (Cuevas et al., 2013)

```
1  S is the total population of spiders:
2  $N$ is the total number of $n$-dimensional colony members;
3  $N_f$ is the number of females;
4  $N_m$ is the number of males;
5  $N_f = floor[(0.9 - rand \cdot 0.25) \cdot N]$
6  $N_m = N - N_f$;
7  Initialize the female $N_f$ and male $N_m$ members randomly;
8  Calculate the radius of mating (S):
9  while (the stopping criteria is not met) do
10     Calculate the weight of every spider(S);
11     Move females according to the female cooperative operator (S);
12     Move males according to the male cooperative operator (S);
13     Perform the mating operation (S);
```

Pereira et al. (2016) address the tuning of parameters for Support Vector Machines (SVM) due to the computational burden for SVM training step. The authors propose an SSO for feature selection and parameter tuning. SSO is decided to be a suitable approach for the model selection of SVM. Cuevas and Cienfuegos (2014) propose SSO for solving constrained optimization tasks. Simulation and comparisons based on several well studied benchmarks functions and real-world engineering problems demonstrate the effectiveness, efficiency and stability of the proposed method. El-Bages and Elsayed (2017) propose SSO for the solution of the static transmission expansion planning. A DC power flow sub-problem is solved for each network resulting from adding a potential solution developed by the SSO algorithm to the base network. James and Li (2016) propose a new SSO algorithm to solve the Economic Load Dispatch (ELD) problem that is an important part of power system control and operation. Zhou, Zhou, Luo, and Abdel-Basset (2017) propose a simplex method-based SSO algorithm to overcome the converge to local minima problem. James and Li (2015) propose SSO for solving the global optimization problem. The authors carry out parameter sensitivity analysis and develop guidelines for selecting the parameter values. Elsayed, Hegazy, Bendary, and El-Bages (2016) propose a modified SSO algorithm for the solution of the non-convex economic dispatch problem. Kurdi (2018) develops a SSO for hybrid flow shop scheduling with multiprocessor task. The proposed algorithm is verified on benchmark problems that it is competitive with state-of-the-art algorithms. Kavitha, Venkumar, Rajini, and Pitchipoo (2018) propose a SSO method for the solution of flexible job shop scheduling problem. The proposed algorithm achieved 92.33% exactness in SSO strategy contrasted with other optimization process. The algorithm is also observed to reduce the execution time.

3.12. Symbiotic organisms search (SOS)

SOS metaheuristic is proposed by Cheng and Prayogo (2014). SOS mimics the interactive behavior among different species of organisms. Organisms mostly live mutually in a swarm for sustenance and survival. This relationship is defined as symbiosis and it describes relationships between distinct species. In symbiosis, two organisms can be linked together to live and they prefer existing in a beneficial relationship together. The well-known relationships in nature are commensalism, mutualism, and parasitism. Commensalism is a symbiotic relationship between two species in which one can get an advantage and the other one is neutral. In mutualism, both benefit from each other. In parasitism, there is a symbiotic relationship between two species in which one benefits and the other is harmed. Symbiotic connections may improve the odds of survival of a species.

The SOS algorithm improves this behavior and has a population to examine the search space while finding the optimal solution. The initial population of a SOS instance is the ecosystem. Each organism represents a solution for the problem to be optimized. Each organism is related to a fitness value that is a kind of adaptation to the solution of the problem. In SOS, new generation is controlled by imitating the biological interaction of two organisms in the ecosystem. The phases of the algorithm are commensalism, mutualism, and parasitism.

In the mutualism phase, let X_i be an organism paired with the ith solution ($i = 1, 2, \ldots, n$) where n is the number of organisms. X_j is another randomly selected organism from the ecosystem to pair with X_i. Solutions (fitness values) for X_i and X_j are computed based on formulae (66) and (67).

$$X_{inew} = X_i + rand(0, 1) * (X_{best} + Mutual\_Vector * BF_1) \tag{66}$$

$$X_{jnew} = X_j + rand(0, 1) * (X_{best} + Mutual\_Vector * BF_2) \tag{67}$$

$$Mutual\_Vector = \left(\frac{X_i + X_j}{2}\right) \tag{68}$$

Parameters BF_1 and BF_1 are selected as either 1 or 2. These coefficients are the levels of the benefit of each organism. Eq. (68) is the “*Mutual_Vector*” that defines the characteristics between organism X_i and X_j. The X_{best} is assumed to be the highest adaptation degree of the ecosystem. Therefore, it is the target value for the fitness evaluations of organisms. Commensalism mimics a relationship of a remora fish and sharks. The remora eats food leftovers of a shark.

A randomly selected X_j interacts with X_i that attempts to benefit from this relationship. X_j does not benefit or suffer from this process. A new solution X_i is evaluated with respect to the commensal symbiosis between X_i and X_j (Eq. (69)).

$$X_{inew} = X_i + rand(-1, 1) * (X_{best} - X_j) \tag{69}$$

$X_{best} - X_j$ is the advantage parameter produced by X_j for X_i to the highest degree in the current organism in the ecosystem.

In parasitism, X_i is used for the generation of the “*Parasite_Vector*”. For each selected X_i, randomly selected dimensions are modified. Randomly selected organism, X_j, is a host to the parasite vector. Parasite_Vector tries to swap X_j. The fitness values of both organisms are examined. In case Parasite_Vector has a better fitness value, it will swap organism X_j in the population. Otherwise, X_j will resist to the parasite and the Parasite_ Vector will not exist any longer. In Algorithm 12, the pseudocode of the SOS is presented.

Algorithm 12. Symbiotic Organisms Search Algorithm (Cheng & Prayogo, 2014)

```
1 Initialize the population;
2 while termination criterion is not met do
3     Mutualism;
4     Commensalism;
5     Parasitism;
```

Cheng, Prayogo, and Tran (2015) introduce novel discrete symbiotic organisms search for the solution of multiple resources leveling in project scheduling. Tejani, Savsani, and Patel (2016) propose a modified SOS algorithm by introducing adaptive benefit factors in the basic SOS algorithm. The proposed SOS algorithms consider effective combinations of adaptive benefit factors to lay down a good balance between exploration and exploitation of the search space. The results verify that the SOS algorithm is reliable and efficient than the classical SOS and other examined algorithms. Tran, Cheng, and Prayogo (2016) introduce a Multiple Objective SOS (MOSOS) algorithm to solve multiple work shifts problems. The experimental results verify that MOSOS is a powerful search and optimization technique in finding the optimization of work shift schedules. Panda and Pani (2016) propose SOS algorithm to formulate multiobjective problems. The proposed algorithm is integrated with adaptive error function to track equality and inequality constraints. Prasad and Mukherjee (2016) propose an SOS algorithm for the solution of the optimal power flow problem of power system. The results verify the potential of the SOS algorithm for solving hard optimization problems. Tejani, Pholdee, Bureerat, and Prayogo (2018) present a multiobjective adaptive SOS for solving truss optimization problems. The mutualism searches by jumping into unvisited parts of the problem and performs a local search of visited sections. A good balance is provided between an exploration and exploitation phases of the algorithm. Adaptive control is incorporated to propose SOS. Vincent, Redi, Yang, Ruskartina, and Santosa (2017) propose the SOS algorithm for solving the capacitated vehicle routing problem. The problem is a well-known discrete optimization problem for deciding the routes for a set of vehicles serving a set of points with a minimal total routing cost. Ezugwu, Adewumi, and Frıˇncu (2017) present develop a SOS algorithm with SA to solve the NP-Hard traveling salesman problem.

3.13. Teaching-learning-based optimization (TLBO)

TLBO is a population-based metaheuristic algorithm proposed by Rao et al. (2011). The population consists of a group of learners (sample solutions) and a teacher/trainer in a TBLO classroom (population). The first phase of TLBO is the “*Teacher*” Phase and the second phase is the “*Learner*” Phase. TLBO algorithm is a stochastic swarm intelligence algorithm. TLBO has an iterative evolution process that is similar to classical evolutionary algorithms. The lack of algorithm-specific parameters, rapid convergence and easy implementation of TLBO have attracted the attention of researchers. This new method has been applied to engineering design optimization problems (Rao & Patel, 2012; Rao, Savsani, & Vakharia, 2012). Zou, Chen, and Xu (2019) provide a comprehensive survey of prominent TLBO variants and its recent applications and theoretical analysis and detailed information on TLBO can be found in this survey.

Learners in the classroom can obtain knowledge through interaction with a teacher or their classmates. TLBO is based on this simple training model. The best learner is employed as a teacher and he is the most knowledgeable person in the population. The teacher spreads information to learners, which is an exploitation technique commonly used in many classical algorithms. This training process improves the knowledge level (i.e., the overall fitness) of the class. The teacher

Table 2
Comparison of new generation metaheuristics.

Acyronym	#parameters	Stages involving exploration and exploitation	The availability of hybridization	the availability of local search mechanisms
ABC	high	scout dance, food evaluation, travelling	✓	✓
BFO	high	replication, chemotaxis, dispersal, swarming	✗	✓
BA	high	wavelength adjustment, travelling	✓	✗
BBO	high	immigration, mutation, suitability index check	✗	✓
CSA	high	flight, nest selection, removal, and breeding	✓	✗
FA	high	attraction, movement	✓	✗
CSA	high	inertial forces, body interactions and modification	✗	✗
GWA	few	tracking, encircling, attacking, wolf movement	✓	✓
HA	high	pitch adjustment, improvisation, randomization	✗	✓
KH	high	herd and krill movement, attraction, repulsion	✓	✓
SSO	few	reproduction, influence, attraction/dislike, webbing	✗	✓
SOS	few	ecosystem, mutual vector, initial pop. creation	✗	✓
TLBO	parameterless	information speed, learner update, teacher change	✓	✗
WOA	high	encircling, prey search, maneuvering	✓	✓

improves the success of the class with respect to his/her teaching talents. Teacher improves the quality of the learners and when the improvement does not get better, a new and better quality teacher is assigned. The students may require a new higher quality trainer and a new training process can be re-initialized.

M_i is the mean, T_i is the teacher at iteration i and T_i moves M_i towards its own level ($i = 1, 2, \ldots, n$) where n is the number of individuals in the classroom. Therefore, the new mean becomes T_i designated as M_{new}. The new solution is modified with respect to the difference between the current and new mean given by:

$$Difference\_Mean_i := r_i(M_{new} - T_F M_i) \tag{70}$$

where T_F is a *teaching factor* that decides how the mean value will be updated by a teacher, and r_i is a random number in the range of [0, 1]. The value of T_F can be either one or two, which is a heuristic step decided randomly with equal probability as $T_F = \text{round}[1 + \text{rand}(0, 1)\{2\text{–}1\}]$.

This difference changes the current solution according to the expression below:

$$X_{new,i} := X_{old,i} + Difference\_Mean_i \tag{71}$$

Students improve their knowledge by the input from the teacher and the interactions of classmates. A learner interacts with other learners in the classroom randomly. A student learns new things if the other classmate has a better knowledge level. A student is randomly selected from the classroom and this individual trains other randomly selected classmates. If the new individual is better than the former one, it is replaced (see Algorithm 13 for details). The update of the learners for selected two learners where $X_i \neq X_j$ is given as:

$$if\ (X_i < X_j)\ then\ X_{new,i} := X_{old,i} + r_i(X_i - X_j) \tag{72}$$

$$if\ (X_i > X_j)\ then\ X_{new,i} := X_{old,i} + r_i(X_j - X_i) \tag{73}$$

Algorithm 13. Teaching-learning-Based Optimization Algorithm Rao et al. (2011)

```
generate_population(population);
calculate_fitness_of_individuals (population);
for (k:=1 to number_of_generations) do
    for (i:=1 to number_of_individuals) do
        /* Learning from Teacher */
        $T_F$:= round ($r$ + 1);
        $X_{mean}$:= calculate_mean_vector (population);
        $X_{teacher}$:= best_individual (population);
        $X_{new} := X_i + r(X_{teacher} - (T_F X_{mean}))$;
        if ($X_{new}$ is better than $X_i$) then
            $X_i := X_{new}$;
        /* Learning from Classmates */
        $j$:=select_random_individual_from (population);
        if ($X_i$ is better than $X_j$) then
            $X_{i,new} := X_i + r(X_i - X_j)$;
        else
            $X_{i,new} := X_j + r(X_j - X_i)$;
        if ($X_{i,new}$ is better than $X_i$) then
            $X_i := X_{i,new}$;
```

Rao and Patel (2013) propose a TLBO algorithm for the multi-objective optimization of heat exchangers. Plate-fin heat exchanger, shell and tube heat exchanger are considered during the optimization. Kiziloz, Deniz, Dokeroglu, and Cosar (2018) propose novel multiobjective TLBO algorithms with machine learning techniques for the solution of feature subset selection problems. Selecting the minimum number of features while not compromising the accuracy of the results is a multiobjective optimization problem. The authors propose TLBO metaheuristic as a feature subset selection technique and utilize its algorithm-specific parameterless concept. Sevinc and Dokeroglu (2019) propose a novel hybrid TLBO algorithm with extreme learning machines (ELM) for the solution of data classification problems. The proposed algorithm is tested on a set of UCI benchmark datasets. The performance of the algorithm is observed to be competitive for both binary and multiclass data classification problems when compared with state-of-the-art algorithms. Črepinšek, Liu, and Mernik (2012) evaluate the performance of TLBO in a recent survey. The authors report results on TLBO in terms of qualitative and quantitative values. Their results reveal important mistakes about TLBO and provide information for researchers in order to avoid similar mistakes and ensure fair experimental setups of TLBO with other metaheuristics. In a book, non-dominated sorting multiobjective versions of TLBO are explained in detail (Rao, 2016). Constrained/unconstrained, and a multiobjective constrained problem are solved by TLBO. Dokeroglu (2015) proposes a set of new TLBO-based hybrid algorithms to solve quadratic assignment problems. Solution instances are trained with recombination operators and TS optimization engine processes by using exploitation techniques. The algorithms are competitive with other algorithms in literature. Toğan (2012) presents a design procedure employing TLBO techniques for discrete optimization of planar steel frames. Frame examples are inspected to show the suitability of the design procedure. Dede and Ayvaz (2015) propose a TLBO algorithm for the optimization of the size and shape of structures.

3.14. Whale optimization algorithm (WOA)

Mirjalili and Lewis (2016) propose WOA. The WOA is a new metaheuristic inspired by the social behavior of humpback whales. Humpback whales are social animals and use a bubble-net strategy while hunting for fish together. Since whale groups can protect their young easier, humpack whales have developed this group hunting and feeding behavior to their advantage. In this hunting method whales dive under a large group of prey and produce bubbles forcing fish into a bubble-net called bubble-net feeding. This foraging method of humpback whales is used for hunting a large group of small fish or krill since humpback whales have no teeth and a very narrow throat so they can only swallow small prey as a whole (see Fig. 5 for a depiction of this behavior). WOA mathematically models the spiral bubble-net feeding strategy of humpback whales to solve NP-Hard optimization problems. Encircling prey, spiral bubble-net feeding maneuvers, and search for prey are three basic functions used in WOA.

Humpback whales identify the target prey location and start encircling them. The WOA employs many search agents starting each with a random solution. After deciding which search agent has the best solution, the other search agents update their locations towards the global best one as performed by humpback whales. This is given in Eqs. (74) and (75).

$$\overrightarrow{D} = |\overrightarrow{C}\cdot\overrightarrow{X^*}(t) - \overrightarrow{X}(t)| \tag{74}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X^*}(t) - \overrightarrow{A}\cdot\overrightarrow{D} \tag{75}$$

where t represents the iteration steps, $\overrightarrow{A}$ and $\overrightarrow{C}$ are coefficient vectors, X^* is the vector representing the global best solution, $\overrightarrow{X}$ is the position vector and $\cdot$ is an element-by-element multiplication. X^* is iteratively improved as better solutions are discovered at each iteration. $\overrightarrow{A}$ and $\overrightarrow{C}$ are calculated as in Eqs. (76) and (77) respectively.

$$\overrightarrow{A} = 2\overrightarrow{a}\cdot\overrightarrow{r} - \overrightarrow{a} \tag{76}$$

$$\overrightarrow{C} = 2\cdot\overrightarrow{r} \tag{77}$$

where $\overrightarrow{a}$ is iteratively decreased from 2 down to 0 and $\overrightarrow{r}$ is a random vector with size between $[0, 1]$.

Two mechanisms are used for imitating the bubble-net produced by humpback whales: First, a shrinking encircling mechanism is imitated by reducing the value of $\overrightarrow{a}$ in Eq. (76) at each iteration. Second, spiral shaped updating of the position is achieved by calculating the distance between the whale at (X, Y) and prey at (X^*, Y^*). The spiral shaped motion is produced as follows:

$$\overrightarrow{X}(t+1) = D'\cdot e^{bl}\cdot cos(2\pi l) + \overrightarrow{X^*}(t) \tag{78}$$

where $D' = |\overrightarrow{X^*}(t) - \overrightarrow{X}(t)|$ is used to calculate the distance between the prey and the i-th whale, b is a constant parameter for shaping the logarithmic spiral, l is a random number in $[1, 1]$.

The humpback whales can also search for prey randomly by choosing to move towards the position of a random whale instead of the best search solution. Using this method when $|\overrightarrow{A}| > 1$WOA is in the exploration phase and it can perform a global search. The exploration phase can be mathematically modeled as follows:

$$\overrightarrow{D} = |\overrightarrow{C}\cdot\overrightarrow{X_{rand}} - \overrightarrow{X}| \tag{79}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X_{rand}} - \overrightarrow{A}\cdot\overrightarrow{D} \tag{80}$$

where $\overrightarrow{X_{rand}}$ is a random position vector chosen from agent whale population.

WOA randomly generates individuals in the population. These agents can change their locations with respect to either another random agent or the best solution obtained by agents. The a parameter is decreased from two to zero for exploration (when $|a| > 1$) and exploitation (when $|a| < 1$). A random search agent is selected when $|\overrightarrow{A}| > 1$, while the best agent is being selected when $|\overrightarrow{A}| < 1$for updating the position of the agents. Depending on the value of p, WOA is able to act like either a spiral or circular movement. In the end, the WOA is finished by a termination criterion. The details of the WOA is depicted in Algorithm 14.

Algorithm 14. Whale Optimization Algorithm (Mirjalili & Lewis, 2016)

```
Generate random population $X_i$ ($i = 1, 2, ..., n$)
Find fitness value of search agents;
$X*$ = the best agent;
while ($t <$ #iterations) do
    for each agent do
        Update $a, A, C, l$, and $P$
        if ($p < 0.5$) then
            if ($|A| < 1$) then
                Update the position of agents (Equation 74);
            else if ($|A| \geq 1$) then
                Select an agent ($X_{rand}$);
                Update the position of the agent (Equation 80);
        else if ($p \geq 0.5$) then
            Update the position of the agent (Equation 78);
    Validate that search agents go beyond search space;
    Calculate the fitness value of agents;
    Update $X*$ in case a better solution is observed;
    $t++$;
return $X*$;
```

Several WOA-based approaches are presented in the literature in recent years. Kaur and Arora (2018) propose chaotic WOA. Many chaotic maps are proposed as chaotic techniques for setting the parameter(s) of WOA. The proposed algorithm is tested using well-known benchmark functions. The chaotic maps are observed to improve the performance of WOA. Ling, Zhou, and Luo (2017) propose a WOA working with a Lévy flight trajectory. the proposed algorithm is robust, fast, increases the diversity of the population, and avoids search failure caused by premature convergence. El Aziz, Ewees, and Hassanien (2017) examine WOA and Moth-Flame Optimization algorithms to decide the optimal multilevel thresholding for image segmentation. Mafarja and Mirjalili (2018) propose a new WOA based wrapper feature selection algorithm. This work describes two hybrid models to obtain different feature selection techniques based on WOA (Mafarja & Mirjalili, 2017). Jadhav and Gomathi (2018) propose a WOA based data clustering algorithm that tries to determine the optimal centroid for performing the clustering process. The proposed method is experimentally shown to outperform the existing methods. Aljarah, Faris, and Mirjalili (2018) propose a new training WOA to process of artificial neural networks. Prakash and Lakshminarayana (2017) propose a WOA to obtain optimal sizing and placement of capacitors for a typical radial distribution system. Wang, Du, Niu, and Yang (2017) propose a new proposed Multiobjective WOA for wind speed forecasting. Abdel-Basset, Manogaran, El-Shahat, and Mirjalili (2018) propose a WOA combined with a local search method for dealing with the permutation flow shop scheduling problem. Oliva, El Aziz, and Hassanien (2017) introduce a Chaotic WOA for the estimation of solar cells parameters. The approach makes use of chaotic maps to set the parameters of the optimization algorithm. El Aziz, Ewees, Hassanien, Mudhsh, and Xiong (2018) develop a WOA for determining the multilevel thresholding values for image segmentation. Although it is a new metaheuristic, WOA seems to have great potential to attract many researchers due to its small number of parameters to be tuned during optimization.

4. Other recent metaheuristic algorithms

In this section, we give brief information about other recent metaheuristic algorithms that attract relatively less attention in the literature. More than 70 metaheuristic studies are briefly investigated. Most of the metaheuristics that have been proposed for the last 20 years are population-based and nature-inspired. Since these algorithms also show promise, we are inclined to refer to them within this survey. The metaheuristic algorithms presented here are organized with respect to their inspirations. Here, we categorize the work in six inspirational categories: Animal herd-based, animal swarm-based, animal behavior-based, natural process-based, astronomy-based metaheuristics, and metaheuristics that are based on other inspirations.

In nature, animal herds can adopt very complex behaviors in order to tackle intractable problems. Such behavior has inspired several studies. Duman, Uysal, and Alkaya (2012) propose Migrating Birds Optimization metaheuristic based on the flight of birds in *V* formation (Niroomand, Hadi-Vencheh, Şahin, & Vizvári, 2015). The algorithm presents an optimization technique with the birds' energy-saving behavior. Askarzadeh (2016) proposes Crow Search Algorithm (CSA) (Sayed, Hassanien, & Azar, 2019; Wang, Zhang, Cao, & Song, 2018). CSA is a population-based and related to the crows that hide their excess food and retrieve it when it is needed. Yazdani and Jolai (2016) propose Lion's algorithm. The natural inspiration is the explanation of such social behavior of lion herds to algorithmic view helps in exploring (near)-optimal solutions from a large search space. Wang, Deb, Gao, and Coelho (2016) propose Elephant Herding Optimization (EHO) algorithm for global optimization problems. EHO is inspired by the elephants that live together. The male elephants leave the groups when they become adults. The behavior of the elephants can be used as clan updating and separating operators.

Swarms can adopt complex behavior even though the individuals in a swarm are not capable of such capacity alone. Thus, swarm behavior also attracted the attention of many studies. Mirjalili (2016a) proposes swarm intelligence optimization technique Dragonfly Algorithm (Ks & Murugan, 2017). Karaboga (2005) presents a report on the swarms of Honey Bees to optimize the combinatorial problems. Later this report forms the basics elements of his ideas on ABC optimization algorithms. Neshat, Sepidnam, Sargolzaei, and Toosi (2014) provide a survey on Artificial Fish Swarm Algorithm (AFSA) (Li, 2002; Shen, Guo, Wu, & Wu, 2011). FSA simulates the social movements of fish where, the fish live in a colony and have swarm intelligence behaviors. Searching for food, migration, dealing with dangerous conditions and interactions between fish are some methods used by the agents of AFSA. Mirjalili et al. (2017) propose single and multiple objective versions of Salp Swarm Algorithm (SSA) for optimization problems. The inspiration is the swarm behavior of salp during their navigation and food search in oceans (Faris et al., 2018). Mirjalili (2015a) also proposes Ant Lion

Optimizer (ALO). The ALO simulates the hunting mechanism of ant lions. Five main steps of hunting prey (building traps, random walk of ants, entrapment of ants, catching prey, and re-building traps) are employed. Saremi, Mirjalili, and Lewis (2017) propose Grasshopper Optimization Algorithm (GOA). GOA is modeled mathematically to mimic the behavior of grasshopper swarms for solving optimization problems (Mafarja et al., 2018). Pinto, Runkler, and Sousa (2007) propose a Wasp Swarm optimization algorithm to achieve the adaption to changes of dynamic MAX-SAT instances obtained from static problems. Meng, Liu, Gao, and Zhang (2014) propose Chicken Swarm Optimization (CSO). CSO simulates the hierarchical relations in a chicken swarm including roosters, hens, and chicks. CSO extracts the swarm intelligence behavior of the chickens while optimizing the problems. CSO is improved by training the part of chicks from the rooster (Wu, Kong, Gao, Shen, & Ji, 2015). Meng, Gao, Lu, Liu, and Zhang (2016) propose a bio-inspired Bird Swarm Algorithm (BSA) algorithm for optimization. BSA is based on the swarm intelligence of birds and their social behaviors and social interactions. Krishnanand and Ghose (2009) present an exposition of a swarm intelligence algorithm for the optimization of multi-modal functions. The main objective of this algorithm is to ensure the capture of all local maxima of the function.

Animals on many occasions may break down complicated concepts into simple procedural processes and solve them in an iterative manner even as individuals. Such behavior inspired many variations of metaheuristics. Oftadeh, Mahjoob, and Shariatpanahi (2010) propose Hunting Search (HuS) inspired by a set of hunter animals such as wolves, dolphins, and lions. The animals search and catch prey by using encircle, tightening the ring of siege operations. Each animal (i.e., agent) sets its location with respect to the location of other animals. Mucherino and Seref (2007) propose Monkey Search Algorithm that simulates the behavior of monkeys climbing trees for food. The branches of the tree are assumed to be the perturbations of neighboring solutions. Jain, Singh, and Rani (2019) propose Squirrel Search Algorithm. The algorithm simulates the foraging manner of flying squirrels and their way of locomotion known as gliding. Au and Benoit-Bird (2003) propose Dolphin Echolocation metaheuristic (Kaveh & Farhoudi, 2013). Dolphins and some animals use the echolocation for navigation and hunting as a biological sonar. This process is simulated to solve combinatorial problems. Mirjalili (2015b) proposes Moth-Flame Optimization (MFO) algorithm. The algorithm simulates the navigation technique of moths in nature. This method is called transverse orientation. Moths fly by keeping an angle with respect to the moon at night. This is a very effective way of traveling in a straight line for a long distance. Pan (2012) proposes Fruit Fly Optimization Algorithm for the optimization of a function. While the function is being tested repeatedly, the population size and other properties are also examined. Abedinia, Amjady, and Ghasemi (2016) propose an algorithm based on the ability of smell sense of sharks and their movement to the odor sources. The algorithm is simulated how sharks find their prey. Wang (2018) proposes Moth Search algorithm that depends on the characteristics of moths that have been the propensity to follow Lévy flights. The best moth individual becomes the light source in this algorithm. Moths that are located next to the fittest one show an aim to fly around in the form of Lévy flights. Tilahun and Ong (2015) propose an algorithm based on the prey-predator interaction of animals. Random solutions are chosen as predators and prey with the fitness values of the objective function. A prey runs towards the flock of prey with better values and runs away from predators. Wang, Deb, and Coelho (2015) propose Earthworm Optimization Algorithm. The soil is aerated by earthworms with burrowing and enrich the soil with nutrients. There are two reproduction processes of earthworms, one offspring by itself and one or more than one offspring at one time by using nine improved crossover operators. Sharafi, Khanesar, and Teshnehlab (2016) present an algorithm based on the competitive behavior of various creatures to survive in nature. A competition is designed among birds, cats, bees and ants. Shiqin, Jianjun, and Guangxing (2009) propose Dolphin Partner Optimization (DPO) algorithm based on the bionic study on dolphin. Martin and Stephen (2006) propose biologically inspired algorithm Termite. Individual termites addresses the routing problem in a dynamic network topology.

Many of the processes in nature are inherently procedural may produce complex forms and results even without an interference of an outside intelligence. Thus, many such processes become the inspiration for new metaheuristics. Lam and Li (2010) propose Chemical Reaction Optimization (CRO) that simulates the interactions of molecules to obtain low energy stability. Salimi (2015) proposes the Stochastic Fractal Search (SFS) algorithm inspired by the natural phenomenon of growth. Using the diffusion property which is seen regularly in random fractals, the particles in the new algorithm explore the search space effectively. Zheng (2015) proposes Water Wave Optimization (WWO), for global optimization problems. WWO makes use of phenomena of water waves (propagation, refraction, and breaking). It can be used to obtain effective techniques for searching in high-dimensional problem space. Kaveh and Mahdavi (2014) propose Colliding Bodies Optimization (CBO) algorithm (Kaveh & Ghazaan, 2014). CBO is based on one-dimensional collisions of bodies. Each agent is an object/body with mass. After the collision of two agents, each one moves toward different directions with different velocities. This event causes the agents to move toward better positions in the search space. Doğan and Ölmez (2015) propose a new trajectory metaheuristic called Vortex Search (VS) algorithm to optimize numerical functions. The VS algorithm mimics the vortex pattern that is created by the vortical flow of the stirred fluids. The VS algorithm models its search process as a vortex pattern by using an adaptive step size adjustment method in order to provide a good balance between the exploration and exploitation phases. Kaveh and Khayatazad (2012) propose Ray Optimization metaheuristic. A set of particles constitute the variables of an optimization problem and they are assumed to be rays of light. The set of rays refracts and changes the direction of solutions with the law of refraction. This technique provides an efficient way for the particles while exploring the search space. Sadollah, Bahreininejad, Eskandar, and Hamdi (2013) propose Mine Blast Algorithm (MBA). MBA is a population-based algorithm based on the concept of mine bomb explosion. The algorithm is applied to engineering design and constrained optimization problems effectively. The MBA requires a fewer number of fitness evaluations than the other algorithms. A new optimization technique, Water Cycle Algorithm (WCA), is proposed by Eskandar, Sadollah, Bahreininejad, and Hamdi (2012). The concepts of the algorithm are based on the water cycle process and how rivers flow to the sea in real life. Kashan (2015) proposes optics inspired optimization (OIO) algorithm. OIO assumes the surface of the numerical function to be optimized as a reflecting surface. In this model, each peak reflects as a convex mirror and each valley reflects as a concave one. Kaveh and Bakhshpoori (2016) propose Water Evaporation Optimization (WEO) algorithm. WEO is a physically inspired population-based algorithm. WEO simulates the evaporation of water molecules on a solid surface with different wettability. This process can be studied by the simulations of molecular dynamics. Kaveh and Ghazaan (2017) propose Vibrating Particles System (VPS). VPS is inspired by free vibration of single degree of freedom systems with viscous damping. The solutions are assumed to be particles that obtain their equilibrium. Kaveh and Dadras (2017) introduce Thermal Exchange Optimization (TEO) algorithm that is based the cooling law of Newton. The law states that the heat loss rate of a body is proportional to the temperatures difference in its surroundings and the body. Kaveh and Talatahari (2010) propose an algorithm based on principles from physics and mechanics that utilize the Newtonian laws of mechanics and the Coulomb law from electrostatics.

The astronomical behavior of objects also attracted many as an inspirational source in the field. Mirjalili, Mirjalili, and Hatamlou (2016) propose Multi-verse optimizer. The algorithm is modeled depending on concepts of cosmology (black hole, white hole, and wormhole). The mathematical models used in these concepts are used to explore, exploit, and search the space locally. Hatamlou (2013) proposes Black Hole algorithm. Black Hole is a population-based algorithm that is initialized with a random population. The best solution is selected to be the black hole at each iteration of the algorithm. Later, the black hole starts pulling other candidates around it. Muthiah-Nakarajan and Noel (2016) propose Galactic Swarm Optimization (GSO) algorithm that is inspired by the motion of galaxies, stars, and superclusters of galaxies under gravity. GSO iterates in exploration and exploitation cycles to obtain an optimal trade-off between exploration and exploitation phases. Erol and Eksin (2006) propose an algorithm inspired by the theories of the evolution of the universe.

Some of the studies in the field of metaheuristics base on their inspirations to very creative and unanticipatable sources that they would require their own category. Population-based Sine Cosine Optimization Algorithm (SCA) is proposed by Mirjalili (2016b). Based on sine and cosine functions, SCA creates multiple random initial solutions and improves them to fluctuate outwards or towards the best solution using a model. Moghdani and Salimifard (2018) propose Volleyball Premier League (VPL) metaheuristic algorithm that mimics the competition and interaction among volleyball teams in a season. The algorithm simulates the coaching process of a volleyball team. Terms substitution, coaching, and learning are used in the VPL algorithm to solve optimization problems. Cheng, Qin, Chen, and Shi (2016) propose Brain Storm Optimization (BSO) algorithm. BSO mimics the process of human brainstorming. Individuals are grouped and diverged in the search space. Gonçalves, Lopez, and Miguel (2015) present Search Group Algorithm (SGA), to optimize truss structures. The efficiency of SGA is compared with a set of benchmark problems from the literature. Tamura and Yasuda (2011) propose a multi-point search method for 2D continuous optimization problems. The method is based on spiral phenomena called 2D spiral optimization. Yang (2012b) proposes a flower pollination algorithm, inspired by the pollination process of flowers.

Some of other the recent algorithms are Artificial Chemical Reaction Optimization Algorithm (Alatas, 2011), Exchange Market Algorithm (Ghorbani & Babaei, 2014), Group Counseling Optimization (Eita & Fahmy, 2014), Probability-Based Incremental Learning (Dasgupta & Michalewicz, 2013), Gravitational Local Search (Webster & Bernhard, 2003), Central Force Optimization (Formato, 2007), Curved Space Optimization (Moghaddam, Moghaddam, & Cheriet, 2012), Group Search Optimizer (He, Wu, & Saunders, 2006), Interior Search Algorithm (Gandomi, 2014), Soccer League Competition Algorithm (Moosavian & Roodsari, 2014), Seeker Optimization Algorithm (Dai, Chen, Zhu, & Zhang, 2009), Random Forest Algorithm (Amini, Homayouni, Safari, & Darvishsefat, 2018), Tree-Seed Algorithm (Cinar & Kiran, 2018), Social-based algorithm (Ramezani & Lotfi, 2013), and Invasive Weed Optimization (Goli, Tirkolaee, Malmir, Bian, & Sangaiah, 2019).

5. Recent hybrid metaheuristic algorithms

Hybrid metaheuristic algorithms report significant improvements when they are compared with classical versions of the metaheuristic algorithms. It is perceived from recent studies that more efficient behavior and greater flexibility can be provided by hybrid metaheuristic algorithms (Blum et al., 2011). The main goal of hybrid algorithms is to couple the characteristics of different strategies of metaheuristics and benefit from synergy. In this section, we provide information about hybrid and hyperheuristic algorithms implemented with recent metaheuristics.

Blum et al. (2011) present a survey on hybrid metaheuristics. Powerful hybrid algorithms that are developed by combining different optimization algorithms are explained in this manuscript. Exact algorithms are also reported as a part of these hybrid algorithms. Puchinger and Raidl (2005) present another survey on recent methods of combining metaheuristics and exact algorithms. Obtaining the best or near-optimal solutions is the main goal of these algorithms. Wang, Gandomi, Yang, and Alavi (2016) propose a hybrid algorithm with CSA and KH. The proposed algorithm uses update and abandon operators of KH for CSA and provides efficiency. Wang, Cui, Sun, Rahnamayan, and Yang (2017) deal with the premature convergence problem of FA. The authors propose a new version of FA that uses a random attraction model and new search strategies to obtain an efficient exploration and exploitation process. Mafarja and Mirjalili (2017) propose two hybrid WOA for feature selection where they combined SA with WOA. The results present more efficient models. Gupta and Deep (2019) hybridize GW with DE mutation not to stick into local optima. Optimization problems are solved and the proposed hybridized version has potential to find optimal solution. Dokeroglu (2015) proposes a set of hybrid TLBO to solve the quadratic assignment problem where, TLBO runs well in coordination with Robust TS engine while solving this NP-Hard problem.

The hyperheuristics raise the level of generality while concerning with selecting the right (meta)-heuristic at every condition. The hyperheuristic algorithms operate at a higher level of abstraction and control the use of lower level heuristics that will be applied depending on the search space of the solution. Burke et al. (2010, 2009, 2013) present an overview of hyperheuristic algorithms. Cowling, Kendall, and Soubeiga (2000) analyze different hyperheuristics for a real-world personnel scheduling problem. Burke, Kendall, and Soubeiga (2003) examine hyperheuristics and report evaluations on the timetabling problem. Chakhlevitch and Cowling (2008) present comprehensive study on recent developments in hyperheuristics. Dokeroglu and Cosar (2016) propose a parallel hyperheuristic for the quadratic assignment problem. Beyaz, Dokeroglu, and Cosar (2015) propose a hyperheuristic algorithm for the solution of offline 2D bin packing problems.

Elaziz and Mirjalili (2019) develop a hyperheuristic to improve the performance of WOA by using DE algorithm. Damaševičius and Woźniak (2017) present a hyperheuristic using a logistic probability function. The algorithm is implemented by using ABC and KH metaheuristics. Wang and Guo (2013) propose a novel robust hybrid metaheuristic method with BA in order to solve global numerical optimization problems. The performance of the algorithm is observed to be superior to classical population-based metaheuristics. Tawhid and Ali (2017) propose Hybrid GWO and GA to obtain the minimal energy of a molecule. The algorithm is used to stabilize the exploitation and exploration efforts. The genetic mutation operator refrains from the earlier convergence and local optima.

Population-based algorithms that are integrated with local search techniques are some of the well-known and best performing implementations of hybrid algorithms (Talbi, 2009). ParadisEO is a software framework for hybrid metaheuristics to optimize single and multiobjective problems in single- and multi-computer environments (Cahon, Melab, & Talbi, 2004; Tirkolaee, Goli, Hematian, Sangaiah, & Han, 2019).

There are many other hybrid algorithms facilitating the new generation metaheuristics. Our aim is to draw the attention of researchers to hybrid algorithms rather than providing a survey on all of such metaheuristic algorithms. We believe that the studies on hybrid metaheuristic algorithms could provide better, faster, and more elegant solutions to many complex problems by combining the strengths of different metaheuristics. With the advent of a new wave of metaheuristic algorithms, we foresee that a proportional amount of effort should be spent on hybridization in order to evaluate the true benefits of these algorithms.

6. Conclusion and discussion

This part of our survey addresses critical issues about metaheuristics and new suggestions for possible research opportunities and open challenges of nature-inspired population-based optimization algorithms. In order to examine these critical issues, we first attempt to compare our selected new generation metaheuristic algorithms briefly. Table 2 provides a comparison of the algorithms. Four important features of the new generation algorithms are reported in the table. These features are: The amount of parameters that needs to be addressed to efficiently execute the optimization process, the stages where the algorithm can balance the exploration and exploitation efforts whether the algorithms are used in hybrid metaheuristic studies and the availability of local search mechanisms.

Most of the new generation metaheuristic algorithms examined in this study have a large number of parameters, which is a disadvantage for metaheuristic algorithms. In order to achieve high-quality results in acceptable amounts of time, the parameters used by metaheuristic need to be specifically tuned for the optimization task. Research has progressed in order to overcome this disadvantage. Metaheuristic algorithms such as GWA, SSO, SOS, and TLBO aim to use fewer number of parameters. Similarly, the lack of local search mechanisms that can achieve local optima is another critical issue for metaheuristic algorithms. Having such facilities not only forms a basis for understanding and improving the results of an algorithm but also guarantees that every candidate solution would continue to improve during successive iterations. However, it is important to note that although several metaheuristic algorithms lack such facilities, research shows that their practical applications still achieve high-quality results. We foresee that understanding how metaheuristic approaches achieve successful results theoretically will continue to be an open research question in the upcoming years.

Providing a good balance between exploration and exploitation phases of the algorithm is another important criterion for the performance of the metaheuristic algorithms. Table 2 identifies the stages of new generation metaheuristics that involve different alternative techniques to manipulate the balance between exploration and exploitation. It is clear from the table that many of the evolutionary inspired processes provide mechanisms to control this balance. Finally, although the table shows that hybridization is applied to many of the examined algorithms in the literature, it also identifies that a large number of these algorithms has not been evaluated in tandem, uncovering additional potential research.

One of the widely accepted fundamental benefits of metaheuristic algorithms is that they provide mechanisms to solve large or intractable problems in reasonable execution times while the exact algorithms fail to succeed due to time limitations. Moreover, they are easier to implement and there is no need for ground-truth or background information for the optimization problem to be solved. The optimization is performed on a set of randomly initialized solutions by using evolutionary processes/operators.

The past research indicates that many critical issues are affecting the performance of a metaheuristic. Providing good stability between diversification and intensification is one of these concerns. Diversification searches the solution space globally, whereas intensification focuses on the local solution space. Tuning the iterations of exploration and then directing the search to intensification after spending adequate time is not a trivial setting.

Without any requirement for gradient information, metaheuristics can be implemented easier than exact search algorithms. In many cases, parameter tuning has a significant impact on how well metaheuristic algorithms perform on an optimization problem. The tuning of the parameters of metaheuristic algorithms has very similar reasons and/or implications to the problems faced in machine learning. In addition to the attempts that have been made to provide adaptive parameter settings, another intelligent option is to develop parameterless metaheuristic algorithms. However, developing parameterless metaheuristic algorithms is yet an open problem and needs to be thoroughly studied. More information about the experimental methodologies, the statistical evaluations, and parameter tuning of metaheuristics can be found in (Bartz-Beielstein, Chiarandini, Paquete, & Preuss, 2010; Birattari & Kacprzyk, 2009; McGeoch, 2001).

One of the major shortcomings of the metaheuristic algorithms is that they have to estimate the fitness value of each new solution they produce. The performance declines very quickly when the dimensionality of the problem increases. While solving large-scale optimization problems, calculating fitness forms a computational bottleneck and can be a big obstacle if the complexity of the fitness evaluation is very high. Fast evaluation techniques can be an additional alternative for better metaheuristic algorithms and such availability of quick techniques should be evaluated for these conditions, which would significantly increase the performance. The intention of quick calculation here is not only to speed up the process but also increase the probability of obtaining the best solutions faster. Dynamic programming or parallel computation can be a very efficient way of computing the time-consuming fitness value evaluations. There can be good research opportunities for any metaheuristic since the overall performance depends on the number of iterations.

Hybrid metaheuristics algorithms is an emerging technology in this field. Most of the reported hybrid/hyperheuristic algorithms obtain better solutions than classical metaheuristic algorithms. The combination of diverse metaheuristics can lead to new exciting approaches since the hybridization can be used to get the advantage of different metaheuristics. It is important to note here that studies on hyperheuristics aim to be problem independent and usable by non-specialist researchers in this area.

Most of the time, the performance evaluation of metaheuristics is carried out with statistical analysis due to the lack of a theoretical foundation (Chiarandini, Paquete, Preuss, & Ridge, 2007). There is a need to provide more fairground for statistically sound comparison methods. In accordance with the No Free Lunch Theorem, it is not possible to expect a metaheuristic to perform well for all the class of optimization problems (Ho & Pepyne, 2002; Wolpert et al., 1997). omprehensive discussion about the research directions about the scientific rigor of metaheuristics can be found in a study by Sörensen (2015). Črepinšek et al. (2012) recommend some principles for the fair evaluation of metaheuristic algorithms. They present twelve rules for a fair evaluation. The two of the most important rules in this study are: ”Preferring an equal number of fitness evaluations” and ”Examining those problems on which the proposed algorithm performs well”. In addition to these, experiments should be carried out on a wide spectrum of optimization problems. Studies on benchmark tests involving various optimization problems should be established.

The success of proposed metaheuristic algorithms verifies that the number of studies for developing new metaheuristics will continue to increase in the near future. These efforts will progress until some standards are established in this area and only then, the deficiencies can be identified and evaluations of metaheuristics can be performed more objectively. It is also important to note here that the chaotic versions of the recent metaheuristics can obtain impressive results in the field.

In our opinion, sticking into local optima and efforts to get around this problem while exploring the problem space will always be an important area of research. ”Restarting” is one of the current techniques widely used to alleviate this problem. Tracing the previous local optima or intelligent clustering techniques can be promising novel research directions for the solution of the stagnation problem. A more fruitful research direction in metaheuristic algorithms is to improve the interior structures of current metaheuristic algorithms rather than proposing new ones that are similar to existing algorithms. The research should more focus on adaptive operators, stagnation prevention mechanisms, integration with data mining techniques for increasing the exploration ability of the metaheuristics. It is also a clear requirement to have metaheuristics optimization frameworks that enable developers to compare various algorithms fairly.

References

Abdel-Basset, M., Manogaran, G., El-Shahat, D., & Mirjalili, S. (2018). A hybrid whale

optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. *Future Generation Computer Systems, 85*, 129–145.
Abedinia, O., Amjady, N., & Ghasemi, A. (2016). A new metaheuristic algorithm based on shark smell optimization. *Complexity, 21*(5), 97–116.
Agrawal, V., Sharma, H., & Bansal, J. C. (2012). Bacterial foraging optimization: A survey. *Proceedings of the international conference on soft computing for problem solving (SocProS 2011) December 20–22, 2011* (pp. 227–242). Springer.
Al-Betar, M. A., Doush, I. A., Khader, A. T., & Awadallah, M. A. (2012). Novel selection schemes for harmony search. *Applied Mathematics and Computation, 218*(10), 6095–6117.
Alatas, B. (2010). Chaotic harmony search algorithms. *Applied Mathematics and Computation, 216*(9), 2687–2699.
Alatas, B. (2011). Acroa: Artificial chemical reaction optimization algorithm for global optimization. *Expert Systems with Applications, 38*(10), 13170–13180.
Alba, E. (2005). *Parallel metaheuristics: a new class of algorithms*, *Vol. 47*. John Wiley & Sons.
Alba, E., Luque, G., & Nesmachnow, S. (2013). Parallel metaheuristics: Recent advances and new trends. *International Transactions in Operational Research, 20*(1), 1–48.
Alba, E., & Troya, J. M. (1999). A survey of parallel distributed genetic algorithms. *Complexity, 4*(4), 31–52.
Aljarah, I., Faris, H., & Mirjalili, S. (2018). Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Computing, 22*(1), 1–15.
Amini, S., Homayouni, S., Safari, A., & Darvishsefat, A. A. (2018). Object-based classification of hyperspectral data using random forest algorithm. *Geo-spatial Information Science, 21*(2), 127–138.
Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. *Computers & Structures, 169*, 1–12.
Au, W. W., & Benoit-Bird, K. J. (2003). Automatic gain control in the echolocation system of dolphins. *Nature, 423*(6942), 861.
Awasthi, A., & Omrani, H. (2019). A goal-oriented approach based on fuzzy axiomatic design for sustainable mobility project selection. *International Journal of Systems Science: Operations & Logistics, 6*(1), 86–98.
Baghel, M., Agrawal, S., & Silakari, S. (2012). Survey of metaheuristic algorithms for combinatorial optimization. *International Journal of Computer Applications, 58*(19).
Banzhaf, W., Nordin, P., Keller, R. E., & Francone, F. D. (1998). *Genetic programming: An introduction*, *Vol. 1*. San Francisco: Morgan Kaufmann.
Bartz-Beielstein, T., Chiarandini, M., Paquete, L., & Preuss, M. (2010). *Experimental methods for the analysis of optimization algorithms.* Springer.
Basturk, B. (2006). An artificial bee colony (abc) algorithm for numeric function optimization. *IEEE swarm intelligence symposium, Indianapolis, IN, USA, 2006*.
Basu, M., & Chowdhury, A. (2013). Cuckoo search algorithm for economic dispatch. *Energy, 60*, 99–108.
Beyaz, M., Dokeroglu, T., & Cosar, A. (2015). Robust hyper-heuristic algorithms for the offline oriented/non-oriented 2d bin packing problems. *Applied Soft Computing, 36*, 236–245.
Bhandari, A. K., Singh, V. K., Kumar, A., & Singh, G. K. (2014). Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using Kapur's entropy. *Expert Systems with Applications, 41*(7), 3538–3560.
Bhattacharya, A., & Chattopadhyay, P. K. (2010). Hybrid differential evolution with biogeography-based optimization for solution of economic load dispatch. *IEEE Transactions on Power Systems, 25*(4), 1955–1964.
Bianchi, L., Dorigo, M., Gambardella, L. M., & Gutjahr, W. J. (2006). Metaheuristics in stochastic combinatorial optimization: A survey. TechReport: Dalle Molle Institute for Artificial Intelligence.
Bianchi, L., Dorigo, M., Gambardella, L. M., & Gutjahr, W. J. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing, 8*(2), 239–287.
Binitha, S., Sathya, S. S., et al. (2012). A survey of bio inspired optimization algorithms. *International Journal of Soft Computing and Engineering, 2*(2), 137–151.
Birattari, M., & Kacprzyk, J. (2009). *Tuning metaheuristics: A machine learning perspective*, *Vol. 197*. Springer.
Blum, C., Puchinger, J., Raidl, G., Roli, A., et al. (2010). A brief survey on hybrid metaheuristics. *Proceedings of BIOMA*, 3–18.
Blum, C., Puchinger, J., Raidl, G. R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing, 11*(6), 4135–4151.
Bolaji, A. L., Al-Betar, M. A., Awadallah, M. A., Khader, A. T., & Abualigah, L. M. (2016). A comprehensive review: Krill herd algorithm (kh) and its applications. *Applied Soft Computing, 49*, 437–446.
BoussaïD, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences, 237*, 82–117.
Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society, 64*(12), 1695–1724.
Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., & Qu, R. (2009). A survey of hyper-heuristics. Computer Science Technical Report No. NOTTCS-TR-SUB-0906241418-2747, School of Computer Science and Information Technology, University of Nottingham.
Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Woodward, J. R. (2010). A classification of hyper-heuristic approaches. *Handbook of metaheuristics* (pp. 449–468). Springer.
Burke, E. K., Kendall, G., & Soubeiga, E. (2003). A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics, 9*(6), 451–470.
Cahon, S., Melab, N., & Talbi, E.-G. (2004). Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics, 10*(3), 357–380.
Camacho-Villalón, C. L., Dorigo, M., & Stützle, T. (2019). The intelligent water drops algorithm: Why it cannot be considered a novel algorithm. *Swarm Intelligence,* 1–20.
Cantú-Paz, E. (1998). A survey of parallel genetic algorithms. *Calculateurs paralleles, reseaux et systems repartis, 10*(2), 141–171.
Chakhlevitch, K., & Cowling, P. (2008). Hyperheuristics: Recent developments. *Adaptive and multilevel metaheuristics* (pp. 3–29). Springer.
Chandrasekaran, K., & Simon, S. P. (2012). Multi-objective scheduling problem: Hybrid approach using fuzzy assisted cuckoo search algorithm. *Swarm and Evolutionary Computation, 5*, 1–16.
Chen, Y.-P., Li, Y., Wang, G., Zheng, Y.-F., Xu, Q., Fan, J.-H., & Cui, X.-T. (2017). A novel bacterial foraging optimization algorithm for feature selection. *Expert Systems with Applications, 83*, 1–17.
Cheng, M.-Y., & Prayogo, D. (2014). Symbiotic organisms search: A new metaheuristic optimization algorithm. *Computers & Structures, 139*, 98–112.
Cheng, M.-Y., Prayogo, D., & Tran, D.-H. (2015). Optimizing multiple-resources leveling in multiple projects using discrete symbiotic organisms search. *Journal of Computing in Civil Engineering, 30*(3), 04015036.
Cheng, S., Qin, Q., Chen, J., & Shi, Y. (2016). Brain storm optimization algorithm: A review. *Artificial Intelligence Review, 46*(4), 445–458.
Chiarandini, M., Paquete, L., Preuss, M., & Ridge, E. (2007). Experiments on metaheuristics: Methodological overview and open issues.
Cinar, A. C., & Kiran, M. S. (2018). Similarity and logic gate-based tree-seed algorithms for binary optimization. *Computers & Industrial Engineering, 115*, 631–646.
Cowling, P., Kendall, G., & Soubeiga, E. (2000). A hyperheuristic approach to scheduling a sales summit. *International conference on the practice and theory of automated timetabling* (pp. 176–190). Springer.
Črepinšek, M., Liu, S.-H., & Mernik, L. (2012). A note on teaching–learning-based optimization algorithm. *Information Sciences, 212*, 79–93.
Cuevas, E., & Cienfuegos, M. (2014). A new algorithm inspired in the behavior of the social-spider for constrained optimization. *Expert Systems with Applications, 41*(2), 412–425.
Cuevas, E., Cienfuegos, M., ZaldíVar, D., & Pérez-Cisneros, M. (2013). A swarm optimization algorithm inspired in the behavior of the social-spider. *Expert Systems with Applications, 40*(16), 6374–6384.
Dai, C., Chen, W., Zhu, Y., & Zhang, X. (2009). Seeker optimization algorithm for optimal reactive power dispatch. *IEEE Transactions on Power Systems, 24*(3), 1218–1231.
Damaševičius, R., & Woźniak, M. (2017). State flipping based hyper-heuristic for hybridization of nature inspired algorithms. *International conference on artificial intelligence and soft computing* (pp. 337–346). Springer.
Das, S., Biswas, A., Dasgupta, S., & Abraham, A. (2009). Bacterial foraging optimization algorithm: Theoretical foundations, analysis, and applications. *Foundations of Computational Intelligence: Vol. 3*, (pp. 23–55). Springer.
Dasgupta, D. (2012). *Artificial immune systems and their applications.* Springer Science & Business Media.
Dasgupta, D., & Michalewicz, Z. (2013). *Evolutionary algorithms in engineering applications.* Springer Science & Business Media.
Dasgupta, S., Das, S., Abraham, A., & Biswas, A. (2009). Adaptive computational chemotaxis in bacterial foraging optimization: An analysis. *IEEE Transactions on Evolutionary Computation, 13*(4), 919–941.
de Castro, L. N., & Timmis, J. I. (2003). Artificial immune systems as a novel soft computing paradigm. *Soft Computing, 7*(8), 526–544.
Dede, T., & Ayvaz, Y. (2015). Combined size and shape optimization of structures with a new meta-heuristic algorithm. *Applied Soft Computing, 28*, 250–258.
Del Ser, J., Osaba, E., Molina, D., Yang, X.-S., Salcedo-Sanz, S., Camacho, D., ... Herrera, F. (2019). Bio-inspired computation: Where we stand and what's next. *Swarm and Evolutionary Computation*.
Doğan, B., & Ölmez, T. (2015). A new metaheuristic for numerical function optimization: Vortex search algorithm. *Information Sciences, 293*, 125–145.
Dokeroglu, T. (2015). Hybrid teaching–learning-based optimization algorithms for the quadratic assignment problem. *Computers & Industrial Engineering, 85*, 86–101.
Dokeroglu, T., & Cosar, A. (2016). A novel multistart hyper-heuristic algorithm on the grid for the quadratic assignment problem. *Engineering Applications of Artificial Intelligence, 52*, 10–25.
Dokeroglu, T., Sevinc, E., & Cosar, A. (2019). Artificial bee colony optimization for the quadratic assignment problem. *Applied Soft Computing, 76*, 595–606.
Dorigo, M., & Birattari, M. (2010). *Ant colony optimization.* Springer.
Dorigo, M., & Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science, 344*(2–3), 243–278.
Duan, C., Deng, C., Gharaei, A., Wu, J., & Wang, B. (2018). Selective maintenance scheduling under stochastic maintenance quality with multiple maintenance actions. *International Journal of Production Research, 56*(23), 7160–7178.
Dubey, R., Gunasekaran, A., & Sushil Singh, T. (2015). Building theory of sustainable manufacturing using total interpretive structural modelling. *International Journal of Systems Science: Operations & Logistics, 2*(4), 231–247.
Duman, E., Uysal, M., & Alkaya, A. F. (2012). Migrating birds optimization: A new metaheuristic approach and its performance on quadratic assignment problem. *Information Sciences, 217*, 65–77.
Duman, S., Güvenç, U., Sönmez, Y., & Yörükeren, N. (2012). Optimal power flow using gravitational search algorithm. *Energy Conversion and Management, 59*, 86–95.
Durgun, İ., & Yildiz, A. R. (2012). Structural design optimization of vehicle components using cuckoo search algorithm. *Materials Testing, 54*(3), 185–188.
e Silva, M.d. A. C., Coelho, L.d. S., & Lebensztajn, L. (2012). Multiobjective biogeography-based optimization based on predator-prey approach. *IEEE Transactions on Magnetics, 48*(2), 951–954.
Eita, M., & Fahmy, M. (2014). Group counseling optimization. *Applied Soft Computing, 22*, 585–604.

El Aziz, M. A., Ewees, A. A., & Hassanien, A. E. (2017). Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. *Expert Systems with Applications, 83*, 242–256.

El Aziz, M. A., Ewees, A. A., Hassanien, A. E., Mudhsh, M., & Xiong, S. (2018). Multi-objective whale optimization algorithm for multilevel thresholding segmentation. *Advances in soft computing and machine learning in image processing* (pp. 23–39). Springer.

El-Bages, M., & Elsayed, W. (2017). Social spider algorithm for solving the transmission expansion planning problem. *Electric Power Systems Research, 143*, 235–243.

Elaziz, M. A., & Mirjalili, S. (2019). A hyper-heuristic for improving the initial population of whale optimization algorithm. *Knowledge-Based Systems, 172*, 42–63.

Elsayed, W., Hegazy, Y., Bendary, F., & El-Bages, M. (2016). Modified social spider algorithm for solving the economic dispatch problem. *Engineering Science and Technology, An International Journal, 19*(4), 1672–1681.

Emary, E., Zawbaa, H. M., & Hassanien, A. E. (2016). Binary grey wolf optimization approaches for feature selection. *Neurocomputing, 172*, 371–381.

Ergezer, M., Simon, D., & Du, D. (2009). Oppositional biogeography-based optimization. *2009 IEEE international conference on systems, man and cybernetics* (pp. 1009–1014). IEEE.

Erol, O. K., & Eksin, I. (2006). A new optimization method: Big bang–big crunch. *Advances in Engineering Software, 37*(2), 106–111.

Eskandar, H., Sadollah, A., Bahreininejad, A., & Hamdi, M. (2012). Water cycle algorithm–a novel metaheuristic optimization method for solving constrained engineering optimization problems. *Computers & Structures, 110*, 151–166.

Espejo, P. G., Ventura, S., & Herrera, F. (2010). A survey on the application of genetic programming to classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 40*(2), 121–144.

Eusuff, M. M., & Lansey, K. E. (2003). Optimization of water distribution network design using the shuffled frog leaping algorithm. *Journal of Water Resources Planning and Management, 129*(3), 210–225.

Ezugwu, A. E.-S., Adewumi, A. O., & Friˇncu, M. E. (2017). Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem. *Expert Systems with Applications, 77*, 189–210.

Farahani, S. M., Abshouri, A., Nasiri, B., & Meybodi, M. (2011). A gaussian firefly algorithm. *International Journal of Machine Learning and Computing, 1*(5), 448.

Faris, H., Mafarja, M. M., Heidari, A. A., Aljarah, I., Ala'M, A.-Z., Mirjalili, S., & Fujita, H. (2018). An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. *Knowledge-Based Systems, 154*, 43–67.

Fister, I., Fister, I., Jr, Yang, X.-S., & Brest, J. (2013). A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation, 13*, 34–46.

Fister Jr, I., Fister, D., & Yang, X.-S. (2013). A hybrid bat algorithm. arXiv preprint arXiv: 1303.6310.

Fister Jr, I., Yang, X.-S., Fister, I., '& Brest, J. (2012). Memetic firefly algorithm for combinatorial optimization. arXiv preprint arXiv: 1204.5165.

Formato, R. A. (2007). Central force optimization. *Progress in Electromagnetics Research, 77*, 425–491.

Gandomi, A. H. (2014). Interior search algorithm (isa): A novel approach for global optimization. *ISA Transactions, 53*(4), 1168–1183.

Gandomi, A. H., & Alavi, A. H. (2012). Krill herd: a new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation, 17*(12), 4831–4845.

Gandomi, A. H., & Yang, X.-S. (2014). Chaotic bat algorithm. *Journal of Computational Science, 5*(2), 224–232.

Gandomi, A. H., Yang, X.-S., & Alavi, A. H. (2011). Mixed variable structural optimization using firefly algorithm. *Computers & Structures, 89*(23–24), 2325–2336.

Gandomi, A. H., Yang, X.-S., & Alavi, A. H. (2013). Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Engineering with Computers, 29*(1), 17–35.

Gandomi, A. H., Yang, X.-S., Alavi, A. H., & Talatahari, S. (2013). Bat algorithm for constrained optimization tasks. *Neural Computing and Applications, 22*(6), 1239–1255.

Gandomi, A. H., Yang, X.-S., Talatahari, S., & Alavi, A. H. (2013). Firefly algorithm with chaos. *Communications in Nonlinear Science and Numerical Simulation, 18*(1), 89–98.

Gao, W.-F., & Liu, S.-Y. (2012). A modified artificial bee colony algorithm. *Computers & Operations Research, 39*(3), 687–697.

Geem, Z. W. (2006). Optimal cost design of water distribution networks using harmony search. *Engineering Optimization, 38*(03), 259–277.

Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation, 76*(2), 60–68.

Gharaei, A., Hoseini Shekarabi, S. A., & Karimi, M. (2019). Modelling and optimal lot-sizing of the replenishments in constrained, multi-product and bi-objective epq models with defective products: Generalised cross decomposition. *International Journal of Systems Science: Operations & Logistics*, 1–13.

Gharaei, A., Karimi, M., & Hoseini Shekarabi, S. A. (2019). Joint economic lot-sizing in multi-product multi-level integrated supply chains: Generalized benders decomposition. *International Journal of Systems Science: Operations & Logistics*, 1–17.

Gharaei, A., Karimi, M., & Shekarabi, S. A. H. (2019). An integrated multi-product, multi-buyer supply chain under penalty, green, and quality control polices and a vendor managed inventory with consignment stock agreement: The outer approximation with equality relaxation and augmented penalty algorithm. *Applied Mathematical Modelling, 69*, 223–254.

Ghorbani, N., & Babaei, E. (2014). Exchange market algorithm. *Applied Soft Computing, 19*, 177–187.

Giri, B., & Bardhan, S. (2014). Coordinating a supply chain with backup supplier through buyback contract under supply disruption and uncertain demand. *International Journal of Systems Science: Operations & Logistics, 1*(4), 193–204.

Giri, B., & Masanta, M. (2018). Developing a closed-loop supply chain model with price and quality dependent demand and learning in production in a stochastic environment. *International Journal of Systems Science: Operations & Logistics,* 1–17.

Glover, F., & Laguna, M. (1998). Tabu search. *Handbook of combinatorial optimization* (pp. 2093–2229). Springer.

Goldberg, D. E. (1989). Genetic algorithms in search. *Optimization, and MachineLearning*.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning, 3*(2), 95–99.

Goli, A., Tirkolaee, E. B., Malmir, B., Bian, G.-B., & Sangaiah, A. K. (2019). A multi-objective invasive weed optimization algorithm for robust aggregate production planning under uncertain seasonal demand. *Computing, 101*(6), 499–529.

Gonçalves, M. S., Lopez, R. H., & Miguel, L. F. F. (2015). Search group algorithm: A new metaheuristic method for the optimization of truss structures. *Computers & Structures, 153*, 165–184.

Gong, W., Cai, Z., Ling, C. X., & Li, H. (2010). A real-coded biogeography-based optimization with mutation. *Applied Mathematics and Computation, 216*(9), 2749–2758.

Guo, L., Wang, G.-G., Gandomi, A. H., Alavi, A. H., & Duan, H. (2014). A new improved krill herd algorithm for global numerical optimization. *Neurocomputing, 138*, 392–402.

Gupta, S., & Deep, K. (2019). Hybrid grey wolf optimizer with mutation operator. *Soft computing for problem solving* (pp. 961–968). Springer.

Hancer, E., Xue, B., Zhang, M., Karaboga, D., & Akay, B. (2018). Pareto front feature selection based on artificial bee colony optimization. *Information Sciences, 422*, 462–479.

Hao, J.-K., & Solnon, C. (2019). Meta-heuristics and artificial intelligence.

Hao, Y., Helo, P., & Shamsuzzoha, A. (2018). Virtual factory system design and implementation: Integrated sustainable manufacturing. *International Journal of Systems Science: Operations & Logistics, 5*(2), 116–132.

Hardy, A. C. (1935). The plankton of the south Georgia whaling grounds and adjacent waters, 1926–1932. *Discovery Rep. 11*, 1–456.

Hassanzadeh, H. R., & Rouhani, M. (2010). A multi-objective gravitational search algorithm. *2010 2nd international conference on computational intelligence, communication systems and networks* (pp. 7–12). IEEE.

Hatamlou, A. (2013). Black hole: A new heuristic optimization approach for data clustering. *Information Sciences, 222*, 175–184.

Hatamlou, A., Abdullah, S., & Nezamabadi-Pour, H. (2012). A combined approach for clustering based on k-means and gravitational search algorithms. *Swarm and Evolutionary Computation, 6*, 47–52.

He, S., Wu, Q., & Saunders, J. (2006). A novel group search optimizer inspired by animal behavioural ecology. *2006 IEEE international conference on evolutionary computation* (pp. 1272–1278). IEEE.

Ho, Y.-C., & Pepyne, D. L. (2002). Simple explanation of the no-free-lunch theorem and its implications. *Journal of Optimization Theory and Applications, 115*(3), 549–570.

Holland, J. H. (1992). Genetic algorithms. *Scientific American, 267*(1), 66–73.

Hoseini Shekarabi, S. A., Gharaei, A., & Karimi, M. (2019). Modelling and optimal lot-sizing of integrated multi-level multi-wholesaler supply chains under the shortage and limited warehouse space: Generalised outer approximation. *International Journal of Systems Science: Operations & Logistics, 6*(3), 237–257.

Hota, P., Barisal, A., & Chakrabarti, R. (2010). Economic emission load dispatch through fuzzy based bacterial foraging algorithm. *International Journal of Electrical Power & Energy Systems, 32*(7), 794–803.

Jadhav, A. N., & Gomathi, N. (2018). Wgc: Hybridization of exponential grey wolf optimizer with whale optimization for data clustering. *Alexandria Engineering Journal, 57*(3), 1569–1584.

Jain, M., Singh, V., & Rani, A. (2019). A novel nature-inspired algorithm for optimization: Squirrel search algorithm. *Swarm and Evolutionary Computation, 44*, 148–175.

James, J., & Li, V. O. (2015). A social spider algorithm for global optimization. *Applied Soft Computing, 30*, 614–627.

James, J., & Li, V. O. (2016). A social spider algorithm for solving the non-convex economic load dispatch problem. *Neurocomputing, 171*, 955–965.

Jati, G. K., et al. (2011). Evolutionary discrete firefly algorithm for travelling salesman problem. *International conference on adaptive and intelligent systems* (pp. 393–403). Springer.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Tech. rep. Technical report-tr06, Erciyes university, engineering faculty, computer.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization, 39*(3), 459–471.

Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (abc) algorithm. *Applied Soft Computing, 8*(1), 687–697.

Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2014). A comprehensive survey: Artificial bee colony (abc) algorithm and applications. *Artificial Intelligence Review, 42*(1), 21–57.

Karaboga, D., & Ozturk, C. (2011). A novel clustering approach: Artificial bee colony (abc) algorithm. *Applied Soft Computing, 11*(1), 652–657.

Karaboga, N. (2009). A new design method based on artificial bee colony algorithm for digital iir filters. *Journal of the Franklin Institute, 346*(4), 328–348.

Kashan, A. H. (2015). A new metaheuristic for optimization: optics inspired optimization (oio). *Computers & Operations Research, 55*, 99–125.

Kaur, G., & Arora, S. (2018). Chaotic whale optimization algorithm. *Journal of Computational Design and Engineering, 5*(3), 275–284.

Kaveh, A., & Bakhshpoori, T. (2016). A new metaheuristic for continuous structural optimization: Water evaporation optimization. *Structural and Multidisciplinary Optimization, 54*(1), 23–43.

Kaveh, A., & Dadras, A. (2017). A novel meta-heuristic optimization algorithm: Thermal exchange optimization. *Advances in Engineering Software, 110*, 69–84.

Kaveh, A., & Farhoudi, N. (2013). A new optimization method: Dolphin echolocation.

Advances in Engineering Software, 59, 53–70.
Kaveh, A., & Ghazaan, M. I. (2014). Enhanced colliding bodies optimization for design problems with continuous and discrete variables. *Advances in Engineering Software, 77*, 66–75.
Kaveh, A., & Ghazaan, M. I. (2017). A new meta-heuristic algorithm: Vibrating particles system. *Scientia Iranica. Transaction A, Civil Engineering, 24*(2), 551.
Kaveh, A., & Khayatazad, M. (2012). A new meta-heuristic method: Ray optimization. *Computers & Structures, 112*, 283–294.
Kaveh, A., & Mahdavi, V. (2014). Colliding bodies optimization: A novel meta-heuristic method. *Computers & Structures, 139*, 18–27.
Kaveh, A., & Talatahari, S. (2010). A novel heuristic optimization method: Charged system search. *Acta Mechanica, 213*(3–4), 267–289.
Kavitha, S., Venkumar, P., Rajini, N., & Pitchipoo, P. (2018). An efficient social spider optimization for flexible job shop scheduling problem. *Journal of Advanced Manufacturing Systems, 17*(02), 181–196.
Kazemi, N., Abdul-Rashid, S. H., Ghazilla, R. A. R., Shekarian, E., & Zanoni, S. (2018). Economic order quantity models for items with imperfect quality and emission considerations. *International Journal of Systems Science: Operations & Logistics, 5*(2), 99–115.
Khan, K., & Sahai, A. (2012). A comparison of ba, ga, pso, bp and lm for training feed forward neural networks in e-learning context. *International Journal of Intelligent Systems and Applications, 4*(7), 23.
Kim, D. H., Abraham, A., & Cho, J. H. (2007). A hybrid genetic algorithm and bacterial foraging approach for global optimization. *Information Sciences, 177*(18), 3918–3937.
Kiziloz, H. E., Deniz, A., Dokeroglu, T., & Cosar, A. (2018). Novel multiobjective tlbo algorithms for the feature subset selection problem. *Neurocomputing, 306*, 94–107.
Kohli, M., & Arora, S. (2018). Chaotic grey wolf optimization algorithm for constrained optimization problems. *Journal of Computational Design and Engineering, 5*(4), 458–472.
Komaki, G., & Kayvanfar, V. (2015). Grey wolf optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time. *Journal of Computational Science, 8*, 109–120.
Krishnanand, K., & Ghose, D. (2009). Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. *Swarm Intelligence, 3*(2), 87–124.
Ks, S. R., & Murugan, S. (2017). Memory based hybrid dragonfly algorithm for numerical optimization problems. *Expert Systems with Applications, 83*, 63–78.
Kumar, V., Chhabra, J. K., & Kumar, D. (2014). Parameter adaptive harmony search algorithm for unimodal and multimodal optimization problems. *Journal of Computational Science, 5*(2), 144–155.
Kurdi, M. (2018). A social spider optimization algorithm for hybrid flow shop scheduling with multiprocessor task. Available at SSRN 3301792.
Lam, A. Y., & Li, V. O. (2010). Chemical-reaction-inspired metaheuristic for optimization. *IEEE Transactions on Evolutionary Computation, 14*(3), 381–399.
Lee, K. S., & Geem, Z. W. (2004). A new structural optimization method based on the harmony search algorithm. *Computers & Structures, 82*(9–10), 781–798.
Lee, K. S., & Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering, 194*(36–38), 3902–3933.
Lee, K. S., Geem, Z. W., Lee, S.-H., & Bae, K.-W. (2005). The harmony search heuristic algorithm for discrete structural optimization. *Engineering Optimization, 37*(7), 663–684.
Lewis, R. (2008). A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum, 30*(1), 167–190.
Li, B., & Jiang, W. (1997). Chaos optimization method and its application. *Control Theory & Applications, 4*.
Li, C., & Zhou, J. (2011). Parameters identification of hydraulic turbine governing system using improved gravitational search algorithm. *Energy Conversion and Management, 52*(1), 374–381.
Li, X.-L. (2002). An optimizing method based on autonomous animats: Fish-swarm algorithm. *Systems Engineering-Theory & Practice, 22*(11), 32–38.
Ling, Y., Zhou, Y., & Luo, Q. (2017). Lévy flight trajectory-based whale optimization algorithm for global optimization. *IEEE Access, 5*, 6168–6186.
Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. *Handbook of metaheuristics* (pp. 320–353). Springer.
Ma, H. (2010). An analysis of the equilibrium of migration models for biogeography-based optimization. *Information Sciences, 180*(18), 3444–3464.
Ma, H., & Simon, D. (2011). Blended biogeography-based optimization for constrained optimization. *Engineering Applications of Artificial Intelligence, 24*(3), 517–525.
Mafarja, M., Aljarah, I., Heidari, A. A., Hammouri, A. I., Faris, H., Ala'M, A.-Z., & Mirjalili, S. (2018). Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems. *Knowledge-Based Systems, 145*, 25–45.
Mafarja, M., & Mirjalili, S. (2018). Whale optimization approaches for wrapper feature selection. *Applied Soft Computing, 62*, 441–453.
Mafarja, M. M., & Mirjalili, S. (2017). Hybrid whale optimization algorithm with simulated annealing for feature selection. *Neurocomputing, 260*, 302–312.
Majumder, A., Laha, D., & Suganthan, P. N. (2018). A hybrid cuckoo search algorithm in parallel batch processing machines with unequal job ready times. *Computers & Industrial Engineering, 124*, 65–76.
Manjarres, D., Landa-Torres, I., Gil-Lopez, S., Del Ser, J., Bilbao, M. N., Salcedo-Sanz, S., & Geem, Z. W. (2013). A survey on applications of the harmony search algorithm. *Engineering Applications of Artificial Intelligence, 26*(8), 1818–1831.
Marques-Silva, J. P., & Sakallah, K. A. (1999). Grasp: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers, 48*(5), 506–521.
Martí, R., Laguna, M., & Glover, F. (2006). Principles of scatter search. *european Journal of operational Research, 169*(2), 359–372.
Martin, R., & Stephen, W. (2006). o: A swarm intelligent routing algorithm for mobile-wireless ad-hoc networks. *Stigmergic optimization* (pp. 155–184). Springer.
McGeoch, C. C. (2001). Experimental analysis of algorithms. *Notices of the AMS, 48*(3), 304–311.
Meng, X., Liu, Y., Gao, X., & Zhang, H. (2014). A new bio-inspired algorithm: Chicken swarm optimization. *International conference in swarm intelligence* (pp. 86–94). Springer.
Meng, X.-B., Gao, X. Z., Lu, L., Liu, Y., & Zhang, H. (2016). A new bio-inspired optimisation algorithm: Bird swarm algorithm. *Journal of Experimental & Theoretical Artificial Intelligence, 28*(4), 673–687.
Mirjalili, S. (2015a). The ant lion optimizer. *Advances in Engineering Software, 83*, 80–98.
Mirjalili, S. (2015b). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89*, 228–249.
Mirjalili, S. (2016a). Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications, 27*(4), 1053–1073.
Mirjalili, S. (2016b). Sca: A sine cosine algorithm for solving optimization problems. *Knowledge-Based Systems, 96*, 120–133.
Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software, 114*, 163–191.
Mirjalili, S., Hashim, S. Z. M., & Sardroudi, H. M. (2012). Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. *Applied Mathematics and Computation, 218*(22), 11125–11137.
Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67.
Mirjalili, S., Mirjalili, S. M., & Hatamlou, A. (2016). Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Computing and Applications, 27*(2), 495–513.
Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.
Mirjalili, S., Mirjalili, S. M., & Yang, X.-S. (2014). Binary bat algorithm. *Neural Computing and Applications, 25*(3–4), 663–681.
Mirjalili, S., Saremi, S., Mirjalili, S. M., & Coelho, L.d. S. (2016). Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization. *Expert Systems with Applications, 47*, 106–119.
Mittal, N., Singh, U., & Sohi, B. S. (2016). Modified grey wolf optimizer for global engineering optimization. *Applied Computational Intelligence and Soft Computing, 2016*, 8.
Mladenović, N., Brimberg, J., Hansen, P., & Moreno-Pérez, J. A. (2007). The p-median problem: A survey of metaheuristic approaches. *European Journal of Operational Research, 179*(3), 927–939.
Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24*(11), 1097–1100.
Moghaddam, F. F., Moghaddam, R. F., & Cheriet, M. (2012). Curved space optimization: a random search based on general relativity theory. arXiv preprint arXiv: 1208.2214.
Moghdani, R., & Salimifard, K. (2018). Volleyball premier league algorithm. *Applied Soft Computing, 64*, 161–185.
Mohan, B. C., & Baskaran, R. (2012). A survey: Ant colony optimization based recent research and implementation on several engineering domain. *Expert Systems with Applications, 39*(4), 4618–4627.
Moosavian, N., & Roodsari, B. K. (2014). Soccer league competition algorithm, a new method for solving systems of nonlinear equations. *International Journal of Intelligence Science, 4*(1), 7–16.
Mucherino, A., & Seref, O. (2007). Monkey search: A novel metaheuristic search for global optimization. *AIP conference proceedings: Vol. 953*, (pp. 162–173). AIP.
Mühlenbein, H. (1992). Parallel genetic algorithms in combinatorial optimization. *Computer science and operations research* (pp. 441–453). Elsevier.
Muthiah-Nakarajan, V., & Noel, M. M. (2016). Galactic swarm optimization: A new global optimization metaheuristic inspired by galactic motion. *Applied Soft Computing, 38*, 771–787.
Nakamura, R. Y., Pereira, L. A., Costa, K. A., Rodrigues, D., Papa, J. P., & Yang, X.-S. (2012). Bba: A binary bat algorithm for feature selection. *2012 25th SIBGRAPI conference on graphics, patterns and images* (pp. 291–297). IEEE.
Nanda, S. J., & Panda, G. (2014). A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary Computation, 16*, 1–18.
Neshat, M., Sepidnam, G., Sargolzaei, M., & Toosi, A. N. (2014). Artificial fish swarm algorithm: A survey of the state-of-the-art, hybridization, combinatorial and indicative applications. *Artificial Intelligence Review, 42*(4), 965–997.
Neumann, F., & Witt, C. (2010). Combinatorial optimization and computational complexity. *Bioinspired computation in combinatorial optimization* (pp. 9–19). Springer.
Niroomand, S., Hadi-Vencheh, A., Şahin, R., & Vizvári, B. (2015). Modified migrating birds optimization algorithm for closed loop layout with exact distances in flexible manufacturing systems. *Expert Systems with Applications, 42*(19), 6586–6597.
Oftadeh, R., Mahjoob, M., & Shariatpanahi, M. (2010). A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search. *Computers & Mathematics with Applications, 60*(7), 2087–2098.
Oliva, D., El Aziz, M. A., & Hassanien, A. E. (2017). Parameter estimation of photovoltaic cells using an improved chaotic whale optimization algorithm. *Applied Energy, 200*, 141–154.
Omkar, S., Senthilnath, J., Khandelwal, R., Naik, G. N., & Gopalakrishnan, S. (2011). Artificial bee colony (abc) for multi-objective design optimization of composite structures. *Applied Soft Computing, 11*(1), 489–499.
Omran, M. G., & Mahdavi, M. (2008). Global-best harmony search. *Applied Mathematics and Computation, 198*(2), 643–656.
Ouaarab, A., Ahiod, B., & Yang, X.-S. (2014). Discrete cuckoo search algorithm for the travelling salesman problem. *Neural Computing and Applications, 24*(7–8), 1659–1669.

Pan, W.-T. (2012). A new fruit fly optimization algorithm: Taking the financial distress model as an example. *Knowledge-Based Systems, 26*, 69–74.
Panda, A., & Pani, S. (2016). A symbiotic organisms search algorithm with adaptive penalty function to solve multi-objective constrained optimization problems. *Applied Soft Computing, 46*, 344–360.
Parejo, J. A., Ruiz-Cortés, A., Lozano, S., & Fernandez, P. (2012). Metaheuristic optimization frameworks: A survey and benchmarking. *Soft Computing, 16*(3), 527–561.
Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine, 22*(3), 52–67.
Passino, K. M. (2010). Bacterial foraging optimization. *International Journal of Swarm Intelligence Research (IJSIR), 1*(1), 1–16.
Pedemonte, M., Nesmachnow, S., & Cancela, H. (2011). A survey on parallel ant colony optimization. *Applied Soft Computing, 11*(8), 5181–5197.
Pereira, D. R., Pazoti, M. A., Pereira, L. A., Rodrigues, D., Ramos, C. O., Souza, A. N., & Papa, J. P. (2016). Social-spider optimization-based support vector machines applied for energy theft detection. *Computers & Electrical Engineering, 49*, 25–38.
Pinto, P. C., Runkler, T. A., & Sousa, J. M. (2007). Wasp swarm algorithm for dynamic max-sat problems. *International conference on adaptive and natural computing algorithms* (pp. 350–357). Springer.
Prakash, D., & Lakshminarayana, C. (2017). Optimal siting of capacitors in radial distribution network using whale optimization algorithm. *Alexandria Engineering Journal, 56*(4), 499–509.
Prasad, D., & Mukherjee, V. (2016). A novel symbiotic organisms search algorithm for optimal power flow of power system with facts devices. *Engineering Science and Technology, An International Journal, 19*(1), 79–89.
Puchinger, J., & Raidl, G. R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. *International work-conference on the interplay between natural and artificial computation* (pp. 41–53). Springer.
Qin, H., Fan, P., Tang, H., Huang, P., Fang, B., & Pan, S. (2019). An effective hybrid discrete grey wolf optimizer for the casting production scheduling problem with multi-objective and multi-constraint. *Computers & Industrial Engineering, 128*, 458–476.
Rabbani, M., Foroozesh, N., Mousavi, S. M., & Farrokhi-Asl, H. (2019). Sustainable supplier selection by a new decision model based on interval-valued fuzzy sets and possibilistic statistical reference point systems under uncertainty. *International Journal of Systems Science: Operations & Logistics, 6*(2), 162–178.
Rabbani, M., Hosseini-Mokhallesun, S. A. A., Ordibazar, A. H., & Farrokhi-Asl, H. (2018). A hybrid robust possibilistic approach for a sustainable supply chain location-allocation network design. *International Journal of Systems Science: Operations & Logistics,* 1–16.
Rajabioun, R. (2011). Cuckoo optimization algorithm. *Applied Soft Computing, 11*(8), 5508–5518.
Ramezani, F., & Lotfi, S. (2013). Social-based algorithm (sba). *Applied Soft Computing, 13*(5), 2837–2856.
Rao, R., & Patel, V. (2012). An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems. *International Journal of Industrial Engineering Computations, 3*(4), 535–560.
Rao, R. V. (2016). Teaching-learning-based optimization algorithm. *Teaching learning based optimization algorithm* (pp. 9–39). Springer.
Rao, R. V., & Patel, V. (2013). Multi-objective optimization of heat exchangers using a modified teaching-learning-based optimization algorithm. *Applied Mathematical Modelling, 37*(3), 1147–1162.
Rao, R. V., Savsani, V. J., & Vakharia, D. (2011). Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design, 43*(3), 303–315.
Rao, R. V., Savsani, V. J., & Vakharia, D. (2012). Teaching-learning-based optimization: An optimization method for continuous non-linear large scale problems. *Information Sciences, 183*(1), 1–15.
Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). Gsa: A gravitational search algorithm. *Information Sciences, 179*(13), 2232–2248.
Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2010). Bgsa: Binary gravitational search algorithm. *Natural Computing, 9*(3), 727–745.
Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2011). Filter modeling using gravitational search algorithm. *Engineering Applications of Artificial Intelligence, 24*(1), 117–122.
Rashedi, E., Rashedi, E., & Nezamabadi-pour, H. (2018). A comprehensive survey on gravitational search algorithm. *Swarm and Evolutionary Computation, 41*, 141–158.
Sabri, N. M., Puteh, M., & Mahmood, M. R. (2013). A review of gravitational search algorithm. *International Journal of Advances in Soft Computing & its Applications, 5*(3), 1–39.
Sadollah, A., Bahreininejad, A., Eskandar, H., & Hamdi, M. (2013). Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. *Applied Soft Computing, 13*(5), 2592–2612.
Salimi, H. (2015). Stochastic fractal search: A powerful metaheuristic algorithm. *Knowledge-Based Systems, 75*, 1–18.
Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper optimisation algorithm: Theory and application. *Advances in Engineering Software, 105*, 30–47.
Sarkar, S., & Giri, B. (2018). Stochastic supply chain model with imperfect production and controllable defective rate. *International Journal of Systems Science: Operations & Logistics,* 1–14.
Sayed, G. I., Hassanien, A. E., & Azar, A. T. (2019). Feature selection via a novel chaotic crow search algorithm. *Neural Computing and Applications, 31*(1), 171–188.
Sayyadi, R., & Awasthi, A. (2018a). An integrated approach based on system dynamics and anp for evaluating sustainable transportation policies. *International Journal of Systems Science: Operations & Logistics,* 1–10.
Sayyadi, R., & Awasthi, A. (2018b). A simulation-based optimisation approach for identifying key determinants for sustainable transportation planning. *International Journal of Systems Science: Operations & Logistics, 5*(2), 161–174.
Schaffer, J. D., Whitley, D., & Eshelman, L. J. (1992). Combinations of genetic algorithms and neural networks: A survey of the state of the art. *[Proceedings] COGANN- 92: International workshop on combinations of genetic algorithms and neural networks* (pp. 1–37). IEEE.
Senthilnath, J., Omkar, S., & Mani, V. (2011). Clustering using firefly algorithm: Performance study. *Swarm and Evolutionary Computation, 1*(3), 164–171.
Sevinc, E., & Dokeroglu, T. (2019). A novel hybrid teaching-learning-based optimization algorithm for the classification of data by using extreme learning machines. *Turkish Journal of Electrical Engineering & Computer Sciences, 27*(2), 1523–1533.
Shabani, M., Mirroshandel, S. A., & Asheri, H. (2017). Selective refining harmony search: A new optimization algorithm. *Expert Systems with Applications, 81*, 423–443.
Shah, N. H., Chaudhari, U., & Cárdenas-Barrón, L. E. (2018). Integrating credit and replenishment policies for deteriorating items under quadratic demand in a three echelon supply chain. *International Journal of Systems Science: Operations & Logistics,* 1–12.
Sharafi, Y., Khanesar, M. A., & Teshnehlab, M. (2016). Cooa: Competitive optimization algorithm. *Swarm and Evolutionary Computation, 30*, 39–63.
Shehab, M., Khader, A. T., & Al-Betar, M. A. (2017). A survey on applications and variants of the cuckoo search algorithm. *Applied Soft Computing, 61*, 1041–1059.
Shen, W., Guo, X., Wu, C., & Wu, D. (2011). Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm. *Knowledge-Based Systems, 24*(3), 378–385.
Shiqin, Y., Jianjun, J., & Guangxing, Y. (2009). A dolphin partner optimization. *2009 WRI global congress on intelligent systems: Vol. 1*, (pp. 124–128). IEEE.
Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation, 12*(6), 702–713.
Simon, D., Ergezer, M., Du, D., & Rarick, R. (2011). Markov models for biogeography-based optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 41*(1), 299–306.
Singh, A. (2009). An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. *Applied Soft Computing, 9*(2), 625–631.
Song, X., Tang, L., Zhao, S., Zhang, X., Li, L., Huang, J., & Cai, W. (2015). Grey wolf optimizer for parameter estimation in surface waves. *Soil Dynamics and Earthquake Engineering, 75*, 147–157.
Sörensen, K. (2015). Metaheuristics-the metaphor exposed. *International Transactions in Operational Research, 22*(1), 3–18.
Sörensen, K., Sevaux, M., & Glover, F. (2018). A history of metaheuristics. *Handbook of Heuristics,* 1–18.
Srinivas, M., & Patnaik, L. M. (1994). Genetic algorithms: A survey. *Computer, 27*(6), 17–26.
Storn, R., & Price, K. (1997). Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341–359.
Taillard, É. D., Gambardella, L. M., Gendreau, M., & Potvin, J.-Y. (2001). Adaptive memory programming: A unified view of metaheuristics. *European Journal of Operational Research, 135*(1), 1–16.
Talbi, E.-G. (2009). *Metaheuristics: From design to implementation, Vol. 74*. John Wiley & Sons.
Tamura, K., & Yasuda, K. (2011). Spiral dynamics inspired optimization. *Journal of Advanced Computational Intelligence and Intelligent Informatics, 15*(8), 1116–1122.
Tang, W., Wu, Q., & Saunders, J. (2006). Bacterial foraging algorithm for dynamic environments. *2006 IEEE International conference on evolutionary computation* (pp. 1324–1330). IEEE.
Tawhid, M. A., & Ali, A. F. (2017). A hybrid grey wolf optimizer and genetic algorithm for minimizing potential energy function. *Memetic Computing, 9*(4), 347–359.
Tejani, G. G., Pholdee, N., Bureerat, S., & Prayogo, D. (2018). Multiobjective adaptive symbiotic organisms search for truss optimization problems. *Knowledge-based Systems, 161*, 398–414.
Tejani, G. G., Savsani, V. J., & Patel, V. K. (2016). Adaptive symbiotic organisms search (sos) algorithm for structural design optimization. *Journal of Computational Design and Engineering, 3*(3), 226–249.
Tilahun, S. L., & Ong, H. C. (2015). Prey-predator algorithm: A new metaheuristic algorithm for optimization problems. *International Journal of Information Technology & Decision Making, 14*(06), 1331–1352.
Tirkolaee, E. B., Goli, A., Hematian, M., Sangaiah, A. K., & Han, T. (2019). Multi-objective multi-mode resource constrained project scheduling problem using pareto-based algorithms. *Computing, 101*(6), 547–570.
Toğan, V. (2012). Design of planar steel frames using teaching–learning based optimization. *Engineering Structures, 34*, 225–232.
Tran, D.-H., Cheng, M.-Y., & Prayogo, D. (2016). A novel multiple objective symbiotic organisms search (mosos) for time–cost–labor utilization tradeoff problem. *Knowledge-Based Systems, 94*, 132–145.
TSai, P.-W., Pan, J.-S., Liao, B.-Y., & Chu, S.-C. (2009). Enhanced artificial bee colony optimization. *International Journal of Innovative Computing, Information and Control,*

5(12), 5081–5092.
Tsao, Y.-C. (2015). Design of a carbon-efficient supply-chain network under trade credits. *International Journal of Systems Science: Operations & Logistics, 2*(3), 177–186.
Tuba, M., Subotic, M., & Stanarevic, N. (2011). Modified cuckoo search algorithm for unconstrained optimization problems. *Proceedings of the 5th European conference on European computing conference* (pp. 263–268). World Scientific and Engineering Academy and Society (WSEAS).
Valian, E., Mohanna, S., & Tavakoli, S. (2011). Improved cuckoo search algorithm for global optimization. *International Journal of Communications and Information Technology, 1*(1), 31–44.
Van Laarhoven, P. J., & Aarts, E. H. (1987). Simulated annealing. *Simulated annealing: Theory and applications* (pp. 7–15). Springer.
Vincent, F. Y., Redi, A. P., Yang, C.-L., Ruskartina, E., & Santosa, B. (2017). Symbiotic organisms search and two solution representations for solving the capacitated vehicle routing problem. *Applied Soft Computing, 52*, 657–672.
Walton, S., Hassan, O., Morgan, K., & Brown, M. (2011). Modified cuckoo search: A new gradient free optimisation algorithm. *Chaos, Solitons & Fractals, 44*(9), 710–718.
Wang, C.-M., & Huang, Y.-F. (2010). Self-adaptive harmony search algorithm for optimization. *Expert Systems with Applications, 37*(4), 2826–2837.
Wang, G., & Guo, L. (2013). A novel hybrid bat algorithm with harmony search for global numerical optimization. *Journal of Applied Mathematics, 2013*.
Wang, G., Guo, L., Gandomi, A. H., Cao, L., Alavi, A. H., Duan, H., & Li, J. (2013). Lévy-flight krill herd algorithm. *Mathematical Problems in Engineering, 2013*.
Wang, G., Guo, L., Wang, H., Duan, H., Liu, L., & Li, J. (2014). Incorporating mutation scheme into krill herd algorithm for global numerical optimization. *Neural Computing and Applications, 24*(3–4), 853–871.
Wang, G.-G. (2018). Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Computing*, 1–14.
Wang, G.-G., Deb, S., & Coelho, L. D. S. (2015). Earthworm optimization algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *International Journal of Bio-Inspired Computation, 7*, 1–23.
Wang, G.-G., Deb, S., Gao, X.-Z., & Coelho, L. D. S. (2016). A new metaheuristic optimisation algorithm motivated by elephant herding behaviour. *International Journal of Bio-Inspired Computation, 8*(6), 394–409.
Wang, G.-G., Gandomi, A. H., & Alavi, A. H. (2014). Stud krill herd algorithm. *Neurocomputing, 128*, 363–370.
Wang, G.-G., Gandomi, A. H., Alavi, A. H., & Hao, G.-S. (2014). Hybrid krill herd algorithm with differential evolution for global numerical optimization. *Neural Computing and Applications, 25*(2), 297–308.
Wang, G.-G., Gandomi, A. H., Yang, X.-S., & Alavi, A. H. (2016). A new hybrid method based on krill herd and cuckoo search for global optimisation tasks. *International Journal of Bio-Inspired Computation, 8*(5), 286–299.
Wang, G.-G., Gandomi, A. H., Zhao, X., & Chu, H. C. E. (2016). Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. *Soft Computing, 20*(1), 273–285.
Wang, G.-G., Guo, L., Gandomi, A. H., Hao, G.-S., & Wang, H. (2014). Chaotic krill herd algorithm. *Information Sciences, 274*, 17–34.
Wang, G.-G., Hossein Gandomi, A., & Hossein Alavi, A. (2013). A chaotic particle-swarm krill herd algorithm for global numerical optimization. *Kybernetes, 42*(6), 962–978.
Wang, H., Cui, Z., Sun, H., Rahnamayan, S., & Yang, X.-S. (2017). Randomly attracted firefly algorithm with neighborhood search and dynamic parameter adjustment mechanism. *Soft Computing, 21*(18), 5325–5339.
Wang, H., Wang, W., Zhou, X., Sun, H., Zhao, J., Yu, X., & Cui, Z. (2017). Firefly algorithm with neighborhood attraction. *Information Sciences, 382*, 374–387.
Wang, J., Du, P., Niu, T., & Yang, W. (2017). A novel hybrid system based on a new proposed algorithm-multi-objective whale optimization algorithm for wind speed forecasting. *Applied Energy, 208*, 344–360.
Wang, W., Zhang, Y., Cao, J., & Song, W. (2018). Robust optimization for volume variation in timber processing. *Journal of Forestry Research, 29*(1), 247–252.
Wang, X., Gao, X.-Z., & Zenger, K. (2015). *An introduction to harmony search optimization method.* Springer.
Webster, B., & Bernhard, P. J. (2003). A local search optimization algorithm based on natural principles of gravitation. Tech. rep.
Wei, Y., & Qiqiang, L. (2004). Survey on particle swarm optimization algorithm. *Engineering Science, 5*(5), 87–94.
Wolpert, D. H., Macready, W. G., et al. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67–82.
Wu, D., Kong, F., Gao, W., Shen, Y., & Ji, Z. (2015). Improved chicken swarm optimization. *2015 IEEE international conference on cyber technology in automation, control, and intelligent systems (CYBER)* (pp. 681–686). IEEE.
Yang, X.-S. (2009). Harmony search as a metaheuristic algorithm. *Music-inspired harmony search algorithm* (pp. 1–14). Springer.
Yang, X.-S. (2010a). Firefly algorithm, stochastic test functions and design optimisation. arXiv preprint arXiv: 1003.1409.
Yang, X.-S. (2010b). *Nature-inspired metaheuristic algorithms.* Luniver Press.
Yang, X.-S. (2010c). A new metaheuristic bat-inspired algorithm. *Nature inspired cooperative strategies for optimization (NICSO 2010)* (pp. 65–74). Springer.
Yang, X.-S. (2012a). Bat algorithm for multi-objective optimisation. arXiv preprint arXiv: 1203.6571.
Yang, X.-S. (2012b). Flower pollination algorithm for global optimization. *International conference on unconventional computing and natural computation* (pp. 240–249). Springer.
Yang, X.-S. (2013a). Bat algorithm: literature review and applications. arXiv preprint arXiv: 1308.3900.
Yang, X.-S. (2013b). *Cuckoo search and firefly algorithm: Theory and applications, Vol. 516*. Springer.
Yang, X.-S. (2013). Multiobjective firefly algorithm for continuous optimization. *Engineering with Computers, 29*(2), 175–184.
Yang, X.-S., & Deb, S. (2009). Cuckoo search via lévy flights. *2009 world congress on nature & biologically inspired computing (NaBIC)* (pp. 210–214). IEEE.
Yang, X.-S., & Deb, S. (2010). Engineering optimisation by cuckoo search. arXiv preprint arXiv: 1005.2908.
Yang, X.-S., & Deb, S. (2014). Cuckoo search: Recent advances and applications. *Neural Computing and Applications, 24*(1), 169–174.
Yang, X.-S., & He, X. (2013). Firefly algorithm: recent advances and applications. arXiv preprint arXiv: 1308.3898.
Yang, X.-S., & Hossein Gandomi, A. (2012). Bat algorithm: A novel approach for global engineering optimization. *Engineering Computations, 29*(5), 464–483.
Yang, X.-S., Hosseini, S. S. S., & Gandomi, A. H. (2012). Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect. *Applied Soft Computing, 12*(3), 1180–1186.
Yazdani, M., & Jolai, F. (2016). Lion optimization algorithm (loa): A nature-inspired metaheuristic algorithm. *Journal of Computational Design and Engineering, 3*(1), 24–36.
Yildiz, A. R. (2013). Cuckoo search algorithm for the selection of optimal machining parameters in milling operations. *The International Journal of Advanced Manufacturing Technology, 64*(1–4), 55–61.
Yılmaz, S., & Küçüksille, E. U. (2015). A new modification approach on bat algorithm for solving optimization problems. *Applied Soft Computing, 28*, 259–275.
Yin, S., Nishi, T., & Zhang, G. (2016). A game theoretic model for coordination of single manufacturer and multiple suppliers with quality variations under uncertain demands. *International Journal of Systems Science: Operations & Logistics, 3*(2), 79–91.
Zavala, G. R., Nebro, A. J., Luna, F., & Coello, C. A. C. (2014). A survey of multi-objective metaheuristics applied to structural optimization. *Structural and Multidisciplinary Optimization, 49*(4), 537–558.
Zheng, Y.-J. (2015). Water wave optimization: A new nature-inspired metaheuristic. *Computers & Operations Research, 55*, 1–11.
Zhou, Y., Zhou, Y., Luo, Q., & Abdel-Basset, M. (2017). A simplex method-based social spider optimization algorithm for clustering analysis. *Engineering Applications of Artificial Intelligence, 64*, 67–82.
Zhu, G., & Kwong, S. (2010). Gbest-guided artificial bee colony algorithm for numerical function optimization. *Applied Mathematics and Computation, 217*(7), 3166–3173.
Zou, D., Gao, L., Wu, J., & Li, S. (2010). Novel global harmony search algorithm for unconstrained problems. *Neurocomputing, 73*(16–18), 3308–3318.
Zou, F., Chen, D., & Xu, Q. (2019). A survey of teaching–learning-based optimization. *Neurocomputing, 335*, 366–383.
Zubair, A. F., & Mansor, M. S. A. (2019). Embedding firefly algorithm in optimization of capp turning machining parameters for cutting tool selections. *Computers & Industrial Engineering*.

Dr. Tansel Dökeroğlu He was born in 1969 in Lüleburgaz/Kirklareli. He graduated from the Department of Mechanical Engineering at the Turkish Military Academy in 1991. In 2006 and 2014, he graduated from Middle East Technical University, Department of Computer Engineering, M.Sc., and Ph.D. Between 1991 and 2014, he worked as a software development specialist, database administrator, and project manager at the Turkish General Staff, Land Forces Command and the Ministry of National Defense information processing units. He retired from his position in the Armed Forces with the rank of Colonel in 2014. In 2017, he received the title of Associate Professor in Computer Engineering. He worked as a Research & Development manager in a private company at METU Teknokent. Between years 2013 and 2015, he taught part-time programming languages at the Department of Computer Engineering, METU. Between the years 2015–2018 he worked in THK Computer Engineering Department as an Asst.Prof.Dr. He served as the Head of the Department of Computer Engineering and Vice Dean at the same university. He continues his studies on meta-heuristic parallel algorithms and machine learning areas. He is married and has one daughter.

Dr. Ender Sevinç received his B.S. degree from Electric/Electronical Department in Military Academy in 1991. Then he received his M.S. and Ph.D. degrees from Computer Engineering Department in Middle East Technical University in 2000 and 2009 respectively. As an industrial experience, he finally worked as Simulation Engineer in NATO JFTC in Poland in 2016. Then he resumed his academic career in University of Turkish Aeronautical Association as an Assistant Professor in 2017. His study and publication areas are query optimization, deep learning and genetic algorithms.

Dr. Tayfun Kucukyilmaz graduated from Ankara Science High School in 1995. He earned his B.S. in Computer Science and Engineering at Bilkent University in 2000. He received his M.S. and Ph.D. degrees in Computer Engineering at Bilkent University in 2003 and 2012, respectively. Between 2010 and 2012, he worked as a contracted intern for 3 months at Yahoo! Research, Barcelona. Between 2012 and 2015 he works as Assistant Professor in University of Turkish Aeronautical Association. He is currently working as Assistant Professor in TED University, Computer Engineering Department. His research interests are: Web Search Engines, machine learning, parallel and distributed systems, high performance computing (HPC), Microprocessors (for HPC), and information retrieval.

Dr. Ahmet Cosar got his B.Sc, M.Sc, and Ph.D degrees, all in computer engineering, from Middle East Technical Uni versity (METU), Bilkent University, and University of Minnesota, respectively. He was a faculty member in METU Computer Engineering department between 1996–2018. His research interests are in distributed databases, data min- ing, e-commerce, and web-based software architectures. Dr. Cosar has also worked as a visiting faculty member in University of Sharjah (UAE) and Manas University (Kyrgyzstan) and also taught a course at American University of Central Asia.