A Survey on Meta-heuristic Algorithms for Global Optimization Problems

Abdel Nasser H. Zaied, Mahmoud Ismail and Salwa El- Sayed*

Department of Operations Research Faculty of Computers and Informatics Zagazig University Egypt
Emails: salwaelsayed_93@yahoo.com; mahsabe@yahoo.com; nasserhr@yahoo.com

Abstract

Optimization is a more important field of research. With increasing the complexity of real-world problems, more efficient and reliable optimization algorithms are vital. Traditional methods are unable to solve these problems so, the first choice for solving these problems becomes meta-heuristic algorithms. Meta-heuristic algorithms proved their ability to solve more complex problems and give more satisfying results. In this paper, we introduce the more popular meta-heuristic algorithms and their applications in addition to providing the more recent references for these algorithms.

Keywords: Optimization; Meta-heuristic algorithms; Nature-inspired algorithms

1. Introduction

Optimization is the process of finding the highest or least objective function value for a set of constraints. In other words, optimization means finding the best solution that can minimize or maximize objective function according to the given problem[1]. The definition of the optimization problem is how to get the values of the variables that can achieve the best value of the objective function.

Mathematical optimization is very important because it works better than the traditional method(guess and check), and it is also not time-consuming for solving a particular problem. Mathematical optimization is useful in many fields like scheduling, finance, inventory control, economics, transportation,…etc.

According to[2], there are two methods of optimization. The first is exact methods, which means that a set of instructions is constructed to solve the problem, and if these instructions are followed correctly, the optimal solution would obtain not just the good solution such as linear programming, integer programming, nonlinear programming, branch and bound, etc... While the second is approximate methods which have two types of algorithms, approximate algorithms that allow getting provable solution and heuristic algorithms which allow obtaining a good solution with acceptable performance. Heuristics is derived from a Greek word, which means "to discover," and it also means "rule of thumb." The heuristic is a method that comes from experience and helps you to think through things the such process of trial and error. Heuristic techniques help you to solve optimization problems faster than you will solve the problem by traditional methods. Heuristic techniques don't guarantee an optimal solution to be got but only satisfy immediate goals, and it includes CDS heuristics, NEH heuristics, Gupta's heuristics, etc..

Heuristic algorithms have two types, meta-heuristics, and problem-specific heuristics. Problem-specific heuristics are constructed to solve a particular problem, and the algorithm can't solve any other problem. While Meta-heuristics is a higher-level procedure designed to find the optimal solution(or near-optimal solution) for the

optimization problem. Meta-heuristic algorithms can handle a wide range of problems, unlike traditional methods. There are more of meta-heuristic algorithms like Genetic Algorithm(GA)[3], Particle Swarm Optimization(PSO)[4], Ant Colony Optimization(ACO)[5], Artificial Bee Colony(ABC)[6], Cuckoo Search(CS)[7], Bat Algorithm(BA)[8] and etc.

All meta-heuristic algorithms handle the problem in the same way because it has two essential processes that must be done. The first process, called "exploration," means that the optimization begins with a collection of random solutions that will be combined and changed rapidly and randomly. The main goal in this process is to determine the desired area that can have a solution. The other process called "exploitation," and the goal of this process is to determine the accuracy of the solutions which had got by the exploration process by using the fitness function(the better solution is that has a higher value of the fitness function) and to get the optimal global solution. Meta-heuristic algorithms have more merits, including the following:

-Meta-heuristic algorithms have flexibility in their behavior because they can deal with different types of problems and also consider the problem as an anonymous box and control only the inputs, outputs, and constraints of the given problem.

-Meta-heuristic algorithms also have simplicity because it uses straightforward rules in bundles or swarms.

-Meta-heuristic algorithms can avoid local solutions(local optima avoidance) and get the desired global optimal solution in a few iterations.

-Collective meta-heuristic algorithms such as GA, PSO, and ACO can generate a set of solutions that cooperate to get optimal global solutions for the given problem.

-The convergence speed of these algorithms is faster than traditional methods for solving the problem.

-These algorithms can also handle a large number of variables, in contrast, to traditional methods.

2. Classification of Meta-heuristic Algorithms

According to [9], meta-heuristic algorithms can be divided into five major categories: swarm-based algorithms, nature-inspired algorithms, biogeographic-simulated algorithms, evolutionary algorithms, and physics-based algorithms.

3. Swarm-based Algorithms

The first category is called swarm-based algorithms. The inspiration for these algorithms is based on the collective behavior of social insects such as bees, ants, and also birds. These algorithms depend on the interaction between swarm members and their environments, such as Ant Colony Optimization(ACO), Artificial Bee Colony(ABC), and Particle Swarm Optimization(PSO).

Ant Colony Optimization(ACO)

It was in 1992 that Ant Colony Optimization (ACO) was introduced by Marco Dorigo in his P.h.D. The notation of this algorithm at the start was to solve the traveling salesman problem, and finding the shortest path between two points(cities) was the main goal at this time and was called the ant system. Since then, more publication based on the ant system was introduced in 1998. In 1998, Dorigo presented the first conference on the ACO algorithm[5]. ACO simulates the behavior of real ants during the searching process for food. Each takes a random path looking for its food, and when it obtains the food, it comes back to the colony. During the searching journey, the ant leaves a pheromone, which guides other ants to the food source. After a specific period, the pheromone starts to evaporate. Ants that return to the colony first, will be the ants that take the shortest path to the food source, and then all ants

will follow this path. ACO starts with a random population of ants. Each ant k moves from state x to state y with the following probability:

$$P_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{Z \in allowed_x} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})} \quad (1)$$

Where τ_{xy} is the pheromone amount borrowed for moving from x to y, α>=0 is the influence control parameter τ_{xy}, η_{xy}=1/d_{xy} is the objective function to be minimized, and d_{xy} is the distance between x, y. β>=1 is the influence control parameter η_{xy}. τ_{xz}, η_{xz} represents pull and way for other states moving.

The pheromone is updated according to the following equation:

$$\tau'_{xy} = (1-\rho)\tau_{xy} + \sum_{k} \Delta\tau_{xy}^{k} \quad (2)$$

Where ρ is the evaporation coefficient of pheromone and $\Delta\tau_{xy}^{k}$ is the amount of pheromone by k^{th} an ant and can be computed by using the following equation:

$$\Delta\tau_{xy}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if ant k uses curve } xy \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Where Q is constant and L_k is the cost of k^{th} the ant. This iteration is repeated until the termination condition is satisfied. ACO has more advantages such as pheromone evaporation that prevents optimal local solution and also density of pheromone that can be heavier in the shortest path, which guides other ants to the food source. ACO is used in more algorithms such as ant robotics, and ant mill, and for more applications like combinatorial optimization problems, traveling salesman problems, and knapsack problems. For more information about this algorithm and its applications, the reader can see these articles[10,11,12].

Artificial Bee Colony (ABC)

In 2005, D Karaboga introduced a new metaheuristic algorithm that depends on honey bee behavior called Artificial Bee Colony(ABC) algorithm [6]. ABC simulates the behavior of bees in nature when bees leave a cell and search for a position in a new cell. They don't fly straight, but they congregate on a branch of a tree that is a little far from the old position. Then all bees gathered around the queen, and then the queen sends a group composed of 20-50 bees, which its mission was to discover the best and nearest position for a new cell. This group of bees has more experience than other bees. By using a special dance, each bee in the group return to tell others about a specified position, and whenever the position was more suitable, whenever the dance was fervent. The dance of bees can express direction and distance of position. For simplicity, ABC is composed of four phases to reach the best solution. Phase one of initialization of population is about many food sources and starts by using the following equation: $x_{ij} = x_j^{\min} + rand(0,1)(x_j^{\min} - x_j^{j\min})$ (4)

where rand(0,1) is a function that generates a random number between 0 and 1. Phase two of the employed bees which its mission is to get nectar to cells and to determine the degree of suitability of the current position by using the following equation:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (5)$$

where $\phi(x_{ij} - x_{kj})$ is the step size. *j* and *k* two indices are chosen.

Phase three of onlooker bees which still in the cell and watch the dance of employed bees before choosing the source of food. They update its position by using the following equations:

If (function value >0)

$$\text{Fitness} = \frac{1}{2*Function\ Value+1} \quad (6)$$

else

$$\text{Fitness} = 1 + fabs\left(\frac{1}{Function\ Value}\right)$$

And to determine the degree of selection for each food source by the following equation:

$$p_{ij} = \frac{fit_i}{\sum_{i=1}^{SN} fit_i} \quad (7)$$

Phase four of scout bees which still also in the cell, but it doesn't depend on information from employed bees and fly randomly to discover other sources. To create new sources of food by using the following equation:

$$x_{ij} = x_j^{\min} + rand(0,1)(x_j^{\min} - x_j^{j\min}) \quad (8)$$

Equations of this algorithm can also be found [13]. ABC is simple and can give better results by changing the size of the population. ABC achieved more success in solving intractable problems so, many publications are introduced for enhancement and to get the best results [14,15]. ABC has more applications like electrical engineering, routing, clustering, design, control, etc.

Particle Swarm Optimization (PSO)

Particle Swarm Optimization(PSO) is the most popular population-based optimization algorithm that was introduced by Kennedy and Eberhart in 1995[4]. The most important advantage of this algorithm is that it doesn't require binary encoding for a given population, such as GA. After introducing PSO, it took more attention from researchers for modification and a combination of it with other algorithms. PSO achieved more success in solving complex optimization problems. PSO algorithm simulates the behavior of particle swarms in nature. Each particle has its velocity and position. Parameters of PSO include Pbest, which is the best fitness value that particle reaches until now, gbest, which is the best fitness value in the overall swarm. PSO starts with a random population, and each particle has a random velocity and a random position. Then, calculate the fitness value for each particle. Then update individual best(pbest) and gbest. After that particle updates its velocity by using the following equation:

$$V_{i+1} = V_i + C1*rand()*(p_{best\,i} - position_i) + C2*(g_{best\,i} - position_i) \quad (9)$$

Where $C1\,C2$ are acceleration constants and can be taken as $C1 = C2$ =2, and *rand* is a random number between 0 and 1. And updates its position by using the following equation

$$position_{i+1} = position_i + v_{i+1} \quad (10)$$

After then, calculate the fitness value for a new position and make comparisons among pbest and gbest. This iteration is repeated until the termination criteria are satisfied. PSO proved its efficiency in solving more complicated problems, so; it is used in more applications such as biomedical, design, control, financial, prediction and sensor networks..etc. For more information about this algorithm and its applications, the reader can see these articles[16,17,18].

4. Nature-inspired Algorithms

The second category is called nature-inspired algorithms, from its name, the inspiration of these algorithms is based on nature. Most of the current algorithms are nature-inspired such as Bat Algorithm(BA) and Cuckoo Search(CS).

Bat Algorithm(BA)

In 2010, a population-based metaheuristic algorithm was developed by X.S.Yang[8]. The notation of this algorithm simulates the echolocation behavior of microbats. BA is more similar to PSO in its behavior; even PSO is considered to be a special case from BA. Parameters of BA include frequency fi, pulse rate ri, loudness Li, velocity vi, and position xi. These parameters can be adjusted according to prey distance. BA starts with a random number of bats with a random frequency, pulse rates, and loudness. Each bat has a random velocity and position. Bats change their velocity by using the following equation:

$$f_i = f_{\min} + (f_{\max} - f_{\min})rand, \quad (11)$$

$$v_{i+1} = v_i + (x_i - x^*) f_i \quad (12)$$

Where rand is a random vector$\in$ [0,1], and x* is the current global best position among all positions in the same iteration. Bats change their position by using the following equation:

$$x_{i+1} = x_i + v_{i+1} \quad (13)$$

changing loudness and pulse rates by using the following equations:

$$L_{i+1} = \beta L_i, \quad (14)$$

$$r_{i+1} = r_i^{t=0}[1 - \exp(-ys)] \quad (15)$$

where β, and γ are constants. BA can effectively make a balance between exploration and exploitation processes so, and more combinations are made between this algorithm and other algorithms to get high performance[19]. Applications of BA include industrial optimization problems, discrete optimization problems, etc..

Cuckoo Search (CS)

In 2009, Yang and Deb introduced the Cuckoo Search algorithm[7]. The idea of this algorithm is based on brood parasitism and simulates the behavior of cuckoo birds. During the flying process, some insects or animals go after long paths with random movement. This random movement is called levy flight. There are three bases for this

algorithm. The first is that every cuckoo lays only one egg and selects a random location(nest) for laying. The second is that only nests with high quality will be moved to the second generation. The third is that the number of host nests is constant, and birds of host nests can discover the egg of the cuckoo by probability pa[0,1]. CS algorithm begins with a random population n(host nests) and then calculates the fitness value for each solution in the population by using a fitness function. After then, create a new solution by using levy flight by using the following equation:

$$x_{i+1} = x_i + \alpha \times levy(\lambda), \quad (16)$$

$$levy(\lambda) = \left| \frac{\tau(1+\lambda) \times \sin(\pi\lambda/2)}{\tau((1+\lambda)/2) \times \lambda \times 2^{((\lambda-1)/2)}} \right|^{1/\lambda} \quad (17)$$

Where α represents step size, Γ is the gamma function, and λ is a constant(1 =< λ=<3). And step size can be calculated by the following equation:

$$r * nests[permute1[i][j] - nests[permute2[i][j] \quad (18)$$

wherein the range [0,1] random number, *permute1*and *permute2* are permutation functions in a different row which applied on the *nests'* matrix. The *nest* is a matrix that includes solutions with their variables.

After then, calculate fitness for a new solution and then pick a random net xj from existing host nests. After then compare the fitness of the picked solution with the fitness of the new solution; if the fitness of the new solution is larger, apply replacement between the two solutions. Otherwise, abandon a new solution. Changing values of parameters for CS will influence the output. The more effective values for parameters are n=15 to 40 and pa=0.25 to 0.5 and =1 to 1.5. parameters of CS are few so, the CS algorithm is easy to implement, resulting in, more adaptive and modified versions of this algorithm being introduced[20,21] to get high performance and efficiency. Applications of CS are more, including structural design optimization, unconstrained optimization problems, and multiobjective optimization problems. For more information about this algorithm and its applications, the reader can see these articles[22,23,24].

5. Biogeographic-simulated Algorithms

The third category is called Biogeographic-simulated algorithms. The inspiration for these algorithms is from biological organisms and it has two basic concepts, migration and mutation such as Spotted Hyena Optimizer(SHO)[10] and Grey Wolf Optimizer(GWO)[11].

Spotted Hyena Optimizer(SHO)

In 2017, a new metaheuristic algorithm called Spotted Hyena Optimizer(SHO) was introduced by Gaurav Dhiman and Vijay Kumar[25]. The notation of this algorithm was inspired by hunting behavior and also the social relationship among Spotted hyenas. There are four types of hyena are, spotted hyena, striped hyena, brown hyena, and aardwolf hyena. Every type can vary in size, diet, and behavior. The spotted hyena is the largest type and can hunt prey effectively. For communication, hyenas can make a sound that can help others to get a food source. This algorithm is based on three steps are, searching for prey, encircling prey, and attacking it. Encircling prey can be mathematically modeled by the following:

$$\vec{D}_h = |\vec{B}.\vec{P}_p(x) - \vec{P}(x)| \quad (19)$$

$$\vec{p}(x+1) = \vec{p}_p(x) - \vec{E}.\vec{D}_h \quad (20)$$

Where $\vec{D}_h$ is the distance between spotted hyena and prey, $\vec{p}$ is the spotted hyena position vector, and $\vec{p}_p$ is the prey position vector? B and E are coefficient vectors and can be computed by the following:

$$\vec{B} = 2.r\vec{d}_1 \quad (21)$$

$$\vec{E} = 2\vec{h}.r\vec{d}_2 - \vec{h} \quad (22)$$

$$h = 5 - (iter \times (5/\max_{iter})) \quad (23)$$

Where $iter$ =1,2,3….. $\max_{iter}$

To achieve a good equilibrium between exploration and exploitation processes $\vec{h}$ is decreased from 5 to 0. While the hunting process and updating the position of spotted hyenas can be performed by the following equation:

$$\vec{D}_h = |\vec{B}.\vec{P}_h - \vec{P}_k| \quad (24)$$

$$\vec{P}_k = \vec{P}_h - \vec{E}.\vec{D}_h \quad (25)$$

$$\vec{C}_h = \vec{P}_k + \vec{P}_{k+1} + ... + \vec{P}_{k+N} \quad (26)$$

Where $\vec{p}_h$ is the first spotted hyena position, $\vec{p}_k$ is the other spotted hyena position (k) and N is the spotted hyena number and can be computed by the following:

$$N = count_{nos}(\vec{p}_h, \vec{p}_{h+1}, \vec{p}_{h+2}....(\vec{p}_h + \vec{M})) \quad (27)$$

Where M is a random vector in [0.5, 1], and *nos* is the number of all candidate solutions. While attacking prey by spotted hyena can be performed by the following:

$$\vec{P}(x+1) = \frac{\vec{C}_h}{N} \quad (28)$$

Where $\vec{p}(x+1)$ can we save the best solutions and update positions. We can conclude from the above equations that spotted hyenas can manage the hunting process effectively. SHO is implemented and compared with other algorithms. Results show that SHO can give competitive results and can get global optimum effectively. SHO is developed to solve multiobjective problems[26] and also is combined with simulated annealing for feature selection[27] and also for non-linear constrained problems[28] and other problems[29].

Grey Wolf Optimizer(GWO)

In 2014, a new metaheuristic algorithm was introduced called Grey Wolf Optimizer(GWO)[30]. GWO is inspired by the leadership behavior of grey wolves as well as the technique of hunting. Spotted Hyena Optimizer(SHO) is more similar to GWO in its behavior in hunting because SHO is introduced after GWO. There are four types or levels of grey wolves, the first type is called alpha, which is the leader of the swarm or pack, and its mission is to lead as well as decide hunting, eating time, and wake-up time. The second type is called beta, which is responsible for communicating with other wolves and telling them about the commands of alpha. Beta also can advise alpha and must respect it. The third type is called delta, which is responsible for discovering a place and having information about attacks from others. Delta is the dominant of the fourth type (omega). The fourth type is called omega, and it's the lowest level of grey wolves. Omega must obey all other grey wolves in the pack(alpha, beta, delta). As mentioned before in SHO, GWO also has the same three steps for hunting, searching for prey, encircling, and attacking prey. Encircling prey can be mathematically modeled by the following equations:

$$\vec{D} = |\vec{C}.\vec{X}_p(t) - \vec{X}(t)| \quad (29)$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A}.\vec{D} \quad (30)$$

Where X is the grey wolf position, Xp is the prey position, A and C are coefficient vectors and can be computed by the following equations:

$$\vec{A} = 2\vec{a}.\vec{r}1 - \vec{a} \quad (31)$$

$$\vec{C} = 2.\vec{r}2 \quad (32)$$

Where r1 and r2 are random vectors in [0,1], and components of a are decreased from 2 to 0. While hunting process can be performed by the following:

$$\vec{D}_{\alpha} = |\vec{C}1.\vec{X}_{\alpha} - \vec{X}|,\ \vec{D}_{\beta} = |\vec{C}2.\vec{X}_{\beta} - \vec{X}|,\ \vec{D}_{\delta} = |\vec{C}3.\vec{X}_{\delta} - \vec{X}| \quad (33)$$

$$\vec{X}1 = \vec{X}_{\alpha} - \vec{A}1.(\vec{D}_{\alpha}),\ \vec{X}2 = \vec{X}_{\beta} - \vec{A}2.(\vec{D}_{\beta}),\ \vec{X}3 = \vec{X}_{\delta} - \vec{A}3.(\vec{D}_{\delta}) \quad (34)$$

$$\vec{X}(t+1) = \frac{\vec{X}1 + \vec{X}2 + \vec{X}3}{3} \quad (35)$$

From the above equations, GWO has only two essential parameters a and c, which can be adjusted effectively to get good results. GWO is implemented and tested in benchmark functions. The results show that GWO has a higher performance in solving optimization problems. For more information about the applications of GWO, the reader can see [31,32,33].

6. Evolutionary Algorithms(EAs)

The fourth category is called evolutionary algorithms**.** Evolutionary algorithms(EAs) were introduced by C. Darwin in 1859 when he showed evolution theory in his book[34]. EAs are random metaheuristics that can be stratified to a wide range of complicated problems. The success that was achieved by these algorithms led to take attention of more researchers to study these algorithms and make more developments for them. Evolutionary Algorithms are classified as genetic algorithms, evolution strategies, genetic programming, and evolutionary programming.

Differential Evolution(DE)

In 1995, Storn and Price introduced a new simple population-based metaheuristic algorithm called differential evolution(DE)[35,36]. Because DE belongs to EAs so, its behavior is the same as EAS and GA, as mentioned before but, DE gives a new meaning to crossover or recombination because crossover here does not swap two individuals but, it combines vectors to generate a new one. Parameters of DE include the number of parents(NP), which is the population, differential weight(F), and crossover constant(CR). Changing values of these parameters surely affect the output. To know how this algorithm can solve a problem. The algorithm initialized the population by a collection of random vectors, and then, three random numbers are picked after then; a tentative solution is created by executing a crossover process between three numbers. Recombining three numbers and generating a new solution by the following equation:

$$y_j^i = x^{n1} + F(x^{n2} - x^{n3}) \quad (36)$$

After then, a comparison between new solutions and old solutions is executed. If the new solution is better or equal to the current solution, the current solution is replaced by the new one. Otherwise, the new solution is ignored. The following equation executes the replacement process:

$$z_j^i = \begin{cases} y_j^i & \text{if } rand() \le CR \text{ or } j = J_{rand} \\ x_j^i & \text{otherwise} \end{cases} \quad (37)$$

. This iteration is repeated until the termination criteria are satisfied. There are no fixed values for parameters that guaranteed the best solution to be reached, but the more suitable values after the experiment of DE are NP>=4, F=0.8, and CR= 0.9 or 1 for a fast solution. After introducing differential evolution, more improvements and combinations with other algorithms are presented for their simplicity and efficiency. DE can be used for a wide

range of problems such as multiobjective optimization problems and continuous optimization problems, etc. for more information about this algorithm and its applications, the reader can see these articles[37,38,39,40].

Genetic Algorithm(GA)

The genetic algorithm(GA) is introduced by J. Holland in 1975[3]. GA achieved more success in solving complex problems and combination with other algorithms and obtaining the best results. The original GA uses binary encoding to represent a problem, but now, GA can be used for more forms. At the start, GA begins with a random population and then calculates the fitness function for each individual in the population, after then, selecting individuals that represent parents by using the following selecting methods[41]: Roulette wheel selection, Stochastic universal sampling, Tournament selection, and Rank-based selection.
After then, parents are reproduced using crossover(recombination), which means a change among two individuals, including one-point crossover and two-point crossover, etc. crossover operator should achieve heritability and validity. Crossover operators' forms include[41]: Intermediate Crossover(IX), Geometrical Crossover(GX), Simulated Binary Crossover(SBX), Two Point Crossover(TPX), etc. Besides using mutation, which means a small change, usually one bit. The mutation operator should achieve validity and locality. The mutation operator forms include 1- for linear discrete: Binary representation mutation and Swapping mutation. 2- for sparse trees: Grow and Switch. 3- for real vectors: Uniform random mutation and Normally distributed mutation. After then, the replacement strategy is applied using the following strategies: Generational replacement and Steady-state replacement Figure(1) shows the pseudo-code of the Genetic Algorithm.

```
generate an initial random population
while iteration <= maxiteration
  iteration = iteration + 1
  calculate the fitness of each individual
  select the individuals according to their fitness
  perform crossover with probability pc
  perform mutation with probability pm
  population = selected individuals after
               crossover and mutation
end while
```

Figure 1: pseudo-code for Genetic Algorithm

For more information about this algorithm, the reader can see these articles[42,43,44]. GA has more applications in solving different problems such as constrained and unconstrained optimization problems even though it can be used as a treatment tool [45].

Evolution Strategy(ES)

In 1964, Rechenberg introduced a new Evolutionary algorithm called evolutionary strategy[46,47]. The notation of this algorithm is that it was distinct among population size and also the replacement process for parents and offspring, which is unlike a Genetic Algorithm. They use mutation and replacement and a little use for recombination. There is more than one schema for selecting population size, the first one (1+1) schema in 1973. This schema means that only one individual from parents will be selected to create offspring. If the generated offspring is better than the parent, the replacement process between the offspring and parent will be executed. The population size for this type is equal to one. Another schema for population size was introduced in 1977(μ,λ) which means that a population size equal to λ and a size μ will be selected from λ to create offspring. There are other schemas for managing population size for more information about evolution strategy and its schemas for population size. The reader can see these articles[48,49,50].

7. Physics-based Algorithms

These algorithms are based on the rules of physics. Scientists employ the laws of physics and chemistry to enhance optimization methods. By using these rules, it can transfer to discover the search space of the given problem and obtain an optimal solution. The most popular physics-based algorithm is Simulated Annealing[51].

Simulated Annealing (SA)

It was in the 1980s that the concepts of annealing were introduced by Kirkpatrick, Gelatt, and Vecchi[51,52]. These concepts included two basic steps: the first step is to put the solid at a very high temperature until melting. The second step is to cool the solid by decreasing the temperature to reach a solid state of minimum energy. Simulated Annealing based on an analogy with materials physical annealing. SA enhances the Monte-Carlo algorithm besides using Metropolis acceptance criteria[53]. The mont-Carlo approach can explore the neighborhood area of the current state to optimize function, but it can be confined to local minima. SA can avoid the drawback of Mont-Carlo by using the Metropolis algorithm. Metropolis algorithm based on the Monte0carlo approach in generation solid states sequence, but it can reach to steady-state. It assumes that to reach the solid to steady-state, it must generate a large number of transitions at every temperature. SA uses these techniques to optimize more complex problems. SA has more merits; the main of them is that SA can accept transitions that break down objective functions. The basic steps of SA are shown in figure(2) where i is the current state, k is the number of iterations, T is the temperature, L is the number of transitions, and the objective function is the solid energy. The following equation can update temperature:

$$T_{(k+1)} = \lambda T_{(k)} \qquad (38)$$

SA is simple and efficient for large-scale problems such as large-scale aircraft trajectory problems [54] and in solving traveling salesman problems [55].

Step 1: Initialization i= i_{start} **, k= 0,** $C_k = C_0$**,** L_k **= L0)**

Step 2: Repeat

Step 3: For l = 0 to L_k **do**

- **Generate a solution j from the neighborhood Si of the current solution i;**
- **If f (j) < f (i) then i = j (j becomes the current solution);**
- **Else, j becomes the current solution with probability :**

$$e^{\left(\frac{f(i)-f(j)}{T}\right)}$$

Step 4: k := k+1;

Step 5:Compute(L_k **;** C_k **);**

Step 6: Until C_k **~=0**

Step 7: terminate.

Figure 2: pseudo-code of SA

8. Conclusion

This paper introduces the most well-known meta-heuristic algorithms and their classes .meta-heuristic algorithms able to solve more complex problems such as convex and large-scale problems that traditional methods can't able to

solve it. We attempt to review a set of these algorithms which take more attention from researchers. We also introduce the merits and applications of each algorithm.

References

[1] Rao S. S. and Rao S., *Engineering optimization: theory and practice*: John Wiley & Sons, 2009.

[2] metaheuristics Published by John Wiley & Sons, Inc., Hoboken, New Jersey
Published simultaneously in Canada.
[3] Holland, John H. "Adaptation in natural and artificial systems Ann Arbor." *The University of Michigan Press* 1 (1975): 975.

[4] Kennedy, J., and R. Eberhart. "Particle swarm optimization (PSO)." *Proc. IEEE International Conference on Neural Networks, Perth, Australia*. 1995.

[5] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Washington, DC, USA, 1999, pp. 1470-1477 Vol. 2.

[6] Tereshko, Valery, and Andreas Loengarov. "Collective decision making in honey-bee foraging dynamics." *Computing and Information Systems* 9.3 (2005): 1.

[7] Yang, Xin-She, and Suash, Deb. "Cuckoo search via Lévy flights." *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*. IEEE, 2009.

[8] Yang, Xin-She. "A new metaheuristic bat-inspired algorithm." *Nature-inspired cooperative strategies for optimization (NICSO, 2010)*. Springer, Berlin, Heidelberg, 2010. 65-74.

[9] Kumar, Ajay, and Seema Bawa. "Generalized ant colony optimizer: a swarm-based meta-heuristic algorithm for cloud services execution." Computing (2018): 1-24.

[10] Chowdhury, Sudipta, et al. "A modified ant colony optimization algorithm to solve a dynamic traveling salesman problem: a case study with drones for wildlife surveillance." *Journal of Computational Design and Engineering* 6.3 (2019): 368-386.

[11] Hu, Xinwu, et al. "Improved ant colony optimization for weapon-target assignment." *Mathematical Problems in Engineering* 2018 (2018).

[12] Hlaing, Z. C. S. S., and May Aye Khine. "An ant colony optimization algorithm for solving the traveling salesman problem." *International Conference on Information Communication and Management*. Vol. 16. 2011.

[13] Karaboga, Dervis, and Bahriye Akay. "A comparative study of artificial bee colony algorithm." *Applied mathematics and computation* 214.1 (2009): 108-132.

[14] Aslan, Selcuk. "A Transition Control Mechanism for Artificial Bee Colony (ABC) Algorithm." *Computational intelligence and neuroscience* 2019 (2019).

[15] Alzaqebah, Malek, and Salwani Abdullah. "Artificial bee colony search algorithm for examination timetabling problems." *International Journal of Physical Sciences* 6.17 (2011): 4264-4272.

[16] Abido, M. A. "Optimal power flow using particle swarm optimization." *International Journal of Electrical Power & Energy Systems* 24.7 (2002): 563-571.

[17] Chau, K. W. "Particle swarm optimization training algorithm for ANNs in stage prediction of Shing Mun River." *Journal of Hydrology* 329.3-4 (2006): 363-367.

[18] Zhang, Jing-Ru, et al. "A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training." *Applied mathematics and computation* 185.2 (2007): 1026-1037.

[19] Gao, Ming-Liang, et al. "A novel visual tracking method using bat algorithm." *Neurocomputing* 177 (2016): 612-619.

[20] Ong, Pauline. "Adaptive cuckoo search algorithm for unconstrained optimization." *The Scientific World Journal* 2014 (2014).

[21] Tuba, Milan, Milos Subotic, and Nadezda Stanarevic. "Modified cuckoo search algorithm for unconstrained optimization problems." *Proceedings of the 5th European conference on the European computing conference*. World Scientific and Engineering Academy and Society (WSEAS), 2011.

[22] Valian, Ehsan, et al. "Improved cuckoo search for reliability optimization problems." *Computers & Industrial Engineering* 64.1 (2013): 459-468.

[23] Yang, Xin-She, and Suash Deb. "Multiobjective cuckoo search for design optimization." *Computers & Operations Research* 40.6 (2013): 1616-1624.

[24] Ouaarab, Aziz, Belaïd Ahead, and Xin-She Yang. "Discrete cuckoo search algorithm for the traveling salesman problem." *Neural Computing and Applications* 24.7-8 (2014): 1659-1669.

[25] Dhiman, Gaurav, and Vijay Kumar. "Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications." *Advances in Engineering Software* 114 (2017): 48-70.

[26] Dhiman, Gaurav, and Vijay Kumar. "Multi-objective spotted hyena optimizer: A Multi-objective optimization algorithm for engineering problems." *Knowledge-Based Systems* 150 (2018): 175-197.

[27] Jia, Heming, et al. "Spotted Hyena Optimization Algorithm with Simulated Annealing for Feature Selection." *IEEE Access* (2019).

[28] Dhiman, Gaurav, and Vijay Kumar. "Spotted hyena optimizer for solving complex and non-linear constrained engineering problems." *Harmony Search and Nature Inspired Optimization Algorithms*. Springer, Singapore, 2019. 857-867.

[29] Balasubbareddy, M., Divyanshi Dwivedi, and D. Sathish. "Optimal Power Flow solution using Spotted Hyena Optimization Algorithm."

[30] Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Andrew Lewis. "Grey wolf optimizer." *Advances in engineering software* 69 (2014): 46-61.

[31] Mirjalili, Seyedali, et al. "Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization." *Expert Systems with Applications* 47 (2016): 106-119.

[32] Song, Xianhai, et al. "Grey Wolf Optimizer for parameter estimation in surface waves." *Soil Dynamics and Earthquake Engineering* 75 (2015): 147-157.

[33] Mittal, Nitin, Ravinder Singh, and Balwinder Singh Sohi. "Modified grey wolf optimizer for global engineering optimization." *Applied Computational Intelligence and Soft Computing* 2016 (2016): 8.

[34] Darwin, Charles. *On the Origin of Species using Natural Selection Or the Preservation of Favoured Races in the Struggle for Life*. H. Milford; Oxford University Press, 1859.

[35] Storn, R. "Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces, Technical report." *International Computer Science Institute* 11 (1995).

[36] Storn, Rainer, and Kenneth Price. "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces." *Journal of global optimization* 11.4 (1997): 341-359.

[37] Ketsripongsa, Udompong, et al. "An Improved Differential Evolution Algorithm for Crop Planning in the Northeastern Region of Thailand." *Mathematical and Computational Applications* 23.3 (2018): 40.

[38] Greco, Rita, and Ivo Vanzi. "New few parameters differential evolution algorithm with application to structural identification." *Journal of Traffic and Transportation Engineering (English Edition)* 6.1 (2019): 1-14.

[39] Leon, Miguel, et al. "A Novel Memetic Framework for Enhancing Differential Evolution Algorithms via Combination With Alopex Local Search." *International Journal of Computational Intelligence Systems* 12.2 (2019): 795-808.

[40] Mohamed, Ali Wagdy, Hegazy Zaher Sabry, and Tareq Abd-Elaziz. "Real parameter optimization by an effective differential evolution algorithm." *Egyptian Informatics Journal* 14.1 (2013): 37-53.

[41] Talbi, El-Ghazali. *Metaheuristics: from design to implementation*. Vol. 74. John Wiley & Sons, 2009.

[42] Ayad, A. R., H. A. Awad, and A. A. Yassin. "Parametric analysis for genetic algorithms handling parameters." *Alexandria Engineering Journal* 52.1 (2013): 99-111.

[43] Toledo, Claudio Fabiano Motta, L. Oliveira, and Paulo Morelato França. "Global optimization using a genetic algorithm with a hierarchically structured population." *Journal of Computational and Applied Mathematics* 261 (2014): 341-351.

[44]. Shimin, Liu, and Wang Zhangang. "Genetic Algorithm and its Application in the path-oriented test data automatic generation." *Procedia Engineering* 15 (2011): 1186-1190.

[45] McCall, John. "Genetic algorithms for modeling and optimization." *Journal of Computational and Applied Mathematics* 184.1 (2005): 205-222.

[46].Rechenberg, Ingo. "Cybernetic solution path of an experimental problem." *Royal Aircraft Establishment Library Translation 1122* (1965).

[47] Rechenberg, Ingo. "Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Frommann–Holzboog." (1973).

[48] Hansen, Nikolaus. "The CMA evolution strategy: a comparing review." *Towards a new evolutionary computation*. Springer, Berlin, Heidelberg, 2006. 75-102.

[49] Hansen, Nikolaus. "The CMA evolution strategy: A tutorial." *arXiv preprint arXiv:1604.00772* (2016).

[50] Mezera-Montes, Efrén, and Carlos A. Coello Coello. "A simple multimembered evolution strategy to solve constrained optimization problems." *IEEE Transactions on Evolutionary Computation* 9.1 (2005): 1-17.

[51] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. Science, 220(4598):671, 1983.

[52] S. Kirkpatrick, D. Gelatt, C, and M.P. Vecchi. Optimization by simulated annealing. IBM Research Report RC 9355, Acts of PTRC Summer Annual Meeting, 1982.

[53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculation by fast computing machines. Journal of Chemical Physics, 21(6):1087–1092, 1953.

[54] A. Islami, S. Chaimatanan, and D. Delahaye. Large-scale 4D trajectory planning. In Electronic Navigation Research Institute, editor, Air Traffic Management and Systems II, volume 420 of Lecture Notes in Electrical Engineering, pages 27–47. Springer Japan, 2017.

[55] H. Bayram and R. Sahin. A new simulated annealing approach for traveling salesman problem. Mathematical and Computational Applications, 18(3):313–322, 2013.