International Journal on
"Technical and Physical Problems of Engineering"
(IJTPE)
Published by International Organization of IOTPE

ISSN 2077-3528
IJTPE Journal
www.iotpe.com
ijtpe@iotpe.com

June 2022 | Issue 51 | Volume 14 | Number 2 | Pages 84-97

SHORT OVERVIEW OF ADVANCED METAHEURISTIC METHODS

N. Tohidi [1] R.B. Rustamov [2]

1. Faculty of Computer Engineering, K.N. Toosi University of Technology, Tehran, Iran, n.tohidi@email.kntu.ac.ir
2. EILINK Research and Development Center, Khazar University, Baku, Azerbaijan, r_rustamov@hotmail.com

Abstract- Advanced metaheuristic methods are considered as an important alternative plan for resolving complicated issues that cannot be addressed with deterministic techniques. In some real tasks, finding the global optimum needs too much time, however, finding near-optimal solutions is acceptable for them, considering the cost of achieving global optima. Accordingly, metaheuristic methods are appropriate for these tasks, because their main feature is the ability to detect approximate solutions in an acceptable time. Actually, randomization helps them to get away from local scale to global search. Hence, almost all metaheuristic methods are suitable for global optimization and nonlinear modeling. In other words, metaheuristics demonstrate high-level soft computing approaches, which can be applied on different types of optimization problems, by prototyping the generic model to each issue, requiring few modifications. Also, they are capable of accurately and efficiently finding near-optimal solutions for a search and/or optimization issue. Therefore, they have vast range of applications in engineering systems and artificial intelligence tasks. Their applications, especially in combination with learning algorithms, have been remarkably increased in recent years and several metaheuristic methods have been proposed with interesting results as well. They have some significant advantages, like simple implementation, great performance and being independent of any problem-specific information such as gradient and etc. For complex and/or large problems, metaheuristic methods are often able to suggest a superior trade-off between the result accuracy and the data processing speed. Furthermore, metaheuristics are very flexible, because they can be adapted to the requirements of most real optimization tasks. Some of the most used metaheuristic approaches are inspired from the nature. Main examples of them are evolutionary algorithms. In the present book chapter, our aim is to present a brief review of metaheuristic methods with a focus on evolutionary algorithms and their main branches like multi-objective and many-objective algorithms, and their examples. In this regard, their common applications would be explained and some successful examples will be introduced.

In addition, some of the most fundamental research aspects related to this subject would be presented. Finally, as concluding remarks, some recommendations will be made for selecting the most suitable metaheuristic methods in different applications with considering their features, particularly in metaheuristic optimization for power energy systems.

Keywords: Metaheuristic Methods, Evolutionary Algorithms, Optimization Methods, Search Methods.

1. INTRODUCTION

Metaheuristics are general approaches which specify algorithms that are capable of accurately and efficiently detecting near-optimal solutions for optimization and/or search problems [1]. Indeed, they depute higher-level heuristic algorithms, that can be applied on various optimization issues, by modeling the outline for individual issues. In addition, they need small modifications to be applied in each particular case.

Intrinsically, many of the nowadays real-word optimization and search problems are NP-hard and complex, so solving them requires a lot of efforts. There are some possible reasons for this:
1) Large volumes of data
2) High dimension search spaces
3) Sparse search space because of having hard constraints
1. Multi-objective or multimodal problems
4) Having complex mathematical functions like time-varying problems

The well-known classical approaches that have precise resolution for optimizing different processes, such as dynamic programming, exhaustive search, linear programming and back-tracking, could discover optimal solutions for complex problems. However, usually they are not applicable in real-world problems, as they require a big running time specially in high-dimensional issues. This is a serious problem in practice, because for NP-hard issues when their instance size grows, the time needed for solving the problem increases in a super-polynomial form [1].

In contrast, metaheuristic methods are probabilistic approximation procedures which are able to solve optimization problem efficiently. Like some other optimization approaches, they cannot guarantee to calculate the optimal solution, although, these methods often find near-optimal and reasonable solutions with good-enough quality in acceptable execution time for almost all kinds of optimization problems. Sometimes, depending on the issue and the particular instance, metaheuristic methods can also calculate optimal solutions. Therefore, they are very popular among researchers today as efficient problem solvers [2]. These soft-computing methods are able to work with uncertainty, imprecision, and estimation errors of the data. Besides, metaheuristics can intelligently work with partial information in order to find acceptable solutions for various problems. Additionally, metaheuristic methods commonly allow researchers to meet the resolution delays existed in different applications.

In recent years, a wide scale of applications, ranging from bioinformatics, informatics optimization, engineering, etc., to commercial, industrial, economics, etc., have been worked out by using several metaheuristic methods [2]. In general, the evaluation and results illustrate that most of the best methods in terms of performance for solving complicated practical issues, by considering both the results precision and the method efficiency, are metaheuristic methods.

Many studies have been presented for and applied an important sub-field of metaheuristic methods called evolutionary algorithms. As mentioned previously, metaheuristics are methods that follows out the recognition of the best solution in a problem search space, systematically. Evolutionary algorithms are one of the most used sub-categories of them. Some main examples of these well-known nature-inspired algorithms are: Genetic Algorithms, Particle Swarm Optimization, Ant Colony Optimization, and etc. These methods have different search patterns, as they explore the solution space. Although, they are all efficient and balanced algorithms for both diversification and intensification, which means exploration of the problem search space and exploitation of pre-learned knowledge, respectively.

The rest of the chapter is structured as follows, Section 2 introduces the main concepts and materials of metaheuristic methods. Next, Section 3 studies the evolutionary algorithms and some main examples in each category of these algorithms. A brief explanation about the application of metaheuristic methods in nowadays real-word issues, with a concentration on their application in power energy systems, is presented in Section 4. Finally, section 5 presents a short conclusion about the main points which are raised in this book chapter.

2. MAIN CONCEPTS AND MATERIALS

Metaheuristic methods are upgraded heuristic methods that their goal is to elevate the search space exploration effectively. Hence, they are able to solve the local search problem in a complex search space. More precisely, they are iterative approaches which aim for applying in each stage a class of heuristic components on a possible solution or a class of possible solutions with an intelligent manner.

The preliminary solution(s) for beginning the search process can either be random solution(s), or can be generated by using some external knowledge to commence the process from a premier position than a random point. The operators of the heuristic process alter the previously detected solution(s), to enhance their quality iteratively. In this regard, the solutions are evaluated according to one or more functions that in some cases called fitness function(s). Sometimes, these functions and the overall search mechanism are inspired by modest phenomena, in other words their main idea comes from mechanisms discovered in nature like collective behavior. The "meta" prefix in "metaheuristic" phrase indicates the superior strategy, which formulates a collection of fundamental heuristic methods to progress the search capabilities in the produced technique. Metaheuristic methods use this superior search strategy in a proximity of the existing possible solution(s). The proximity search is a usual approach applied in many optimization approaches.

The neighborhood of a considered solution distinguishes a collection of possible solutions, which are reachable from the current solution using the predefined operator(s). The search process finishes when a particular termination condition happens. Commonly, the termination condition is associated with a specified number of iterations or time limit, a threshold for fitness function(s), in case of finding the best solution(s), or any mixture of the pre-mentioned conditions. At last, the best-founded solution(s) in the search process, are returned.

In general, metaheuristic methods are inspired by various topics, like simulation with biological, physical, chemical, social or semantic phenomena [3]. Also, they could be classified related to the proximity structure or the strategy for capturing the solution(s) that applied. In the following, the most common classifications for metaheuristic methods are investigated.

2.1. Metaheuristic Methods Classification

Up to now, various classification techniques have been suggested for metaheuristic methods. The most common classification method for metaheuristics that was used in related works considered the number of experimental solutions in each iteration. In some other cases, the selected criteria were related to the procedures that were applied in the search process, such as memory usage, constructive methods and etc.

In general, according to the number of solutions which are managed in each iteration of the search, metaheuristic methods are categorized into two classes [1].

2.1.1. Trajectory-Based Metaheuristic Methods

These methods use a single candidate solution and modify it in each step, so that they replace this solution by another one (usually the best one), which has been found in its proximity. This alternative solution is determined by one or more transformation operators or predefined movements. Obviously, this search process is specified using a trajectory in the candidate solutions space in order to solve the issue.

In fact, these methods are often more efficient and quicker than population-based metaheuristic methods, because they consider just a single solution at any time. Therefore, they can detect a local optimal solution very fast. Accordingly, they also named as exploitation-based techniques, that increase intensification within the search space, via promoting previously-determined good quality solutions, locally.

2.1.2. Population-Based Metaheuristic Methods

These methods use a collection of multiple candidate solutions in each iteration. These selected solutions are altered and/or incorporated by some usual instructions. The population-based approach tries to calculate results with higher quality, using the primary attributes of the original ones. Additionally, they recombine some selected solutions.

In each iteration of these methods, some solutions of the collection are substituted by new ones. In most cases, new solutions are the best ones, or in some cases they are selected according to one or more quality-based criteria. One of the main characteristics of these methods is increasing the diversification by applying several candidate solutions in the population. These metaheuristic methods are referred as exploration-based techniques.

One more beneficial classification criteria of metaheuristics are local search-based methods and constructive methods. The first ones are those which exploit a local strategy to enhance quality of solution(s), by searching in their neighborhood; and later ones, use a particular method to generate new one(s), according to the main features of the current solution(s). Further, as another classification it is possible to distinguish between memory-based and memory-less methods [1]:

• Memory-based are those methods which apply a particular structure that use previous information about the search;

• Memory-less are those techniques which do not work with memory, hence in each step the process is not dependent on the former decisions that were taken during the search.

As well as the traditional models, hybrid methods of metaheuristics have also been suggested to enhance the performance of these methods. Some of them are proposed by including certain knowledge related to the given problem. For example, as specific operators or representations dependent on the issue. The other possibility is to make weak hybrids via merging various methods, in order to take advantage of specific attributes of each method to enhance the search accuracy and its efficiency.

The techniques to combine commonly consist other metaheuristic methods, however, weak hybrids can be constructed using a combination of metaheuristic methods and traditional search methods or machine learning algorithms, too. One probable technique is to have a superior method (commonly in real-world applications it is a population-based metaheuristic method) which supervises a considered application, and a collection of dependent metaheuristic methods (in most cases they are trajectory-based methods). Also, an alternative option can be applying various cooperative metaheuristic methods with the same level in hierarchy [4].

In general, the hybrid metaheuristic concept specified a new search paradigm which designates if a considered method supervises the other methods or a quite cooperative design is used, when each of them is performed, and how each of them send their results to others. In many cases, an impressive cooperation is attained via changing some statistical values or a fine collection of solutions [5].

Figure 1 illustrates the optimization methods classification and its path to the sub-branch of evolutionary algorithms [6]. It should be noted, that in many research Population-based methods divided into two sub-fields:

1. Evolutionary Algorithms
2. Swarm-Intelligence Algorithms

here, for simplicity both of them have been considered as one sub-field (Evolutionary Algorithms).

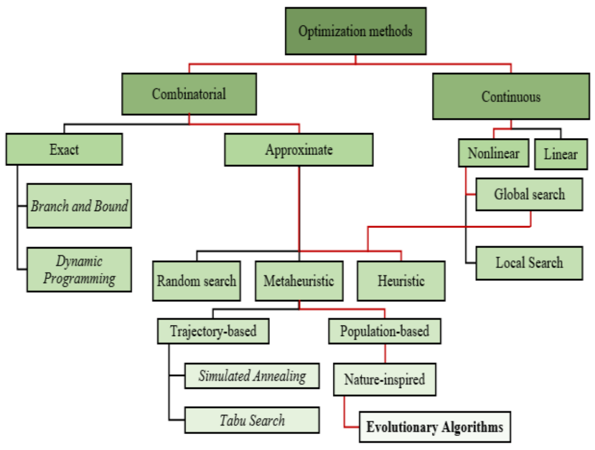


Figure 1. Optimization methods classification [6]

3. EVOLUTIONARY ALGORITHMS

Evolutionary algorithms are a group of nature-inspired algorithms, that have the advantages of some different sub-methods, including population management, iteration, variation and selection. These beneficial algorithms developed based on the primary theory of Darwinian evolution [7]. Commonly, these algorithms are naive in high-levels, but step by step they become more complicated as more domain knowledge is placed in the system.

In order to execute them, some kind of quantitative objective should be set. Actually, this objective would be the metric for measuring success and known as fitness function. Besides, fitness function can be applied as a technique for exiting the process, however, in most cases a function of time is used for this purpose.

Additionally, these algorithms can be designed in a never ending or continuous manner. It is necessary to consider that evolutionary algorithms are usually free of derivatives, which means that they do not need a function to measure amount of change in order to specify the desired result.

When applying evolutionary algorithms for an application, the main aim of the issue can consist of a single-objective or multi(many)-objectives. For multi-objective mode the search approach has to work with the Pareto-fronts. In this case, the algorithm tries to meet all the objectives of the issue to some extent. Here, the discussion continues to interpret these two modes.

3.1. Single-Objective Algorithms

Table 1 represented some single-objective evolutionary algorithms and the related references for each of them.

Table 1. Single-objective evolutionary algorithms examples

Reference	Algorithm name
[8]	Genetic Algorithm (GA)
[9]	Genetic Programming (GP)
[10]	Firefly Algorithm (FA)
[11]	Imperialist Competitive Algorithm (ICA)
[12]	Particle Swarm Optimization (PSO)
[13]	Accelerated Particle Swarm Optimization (APSO)
[14]	Memetic Algorithm (MA)
[15]	Ant Colony Optimization (ACO)
[16]	Flower Pollination Algorithm (FPA)
[17]	Shuffled Frog Leaping (SFL)
[18]	Cat Swarm Optimization (CSO)
[19]	Cuckoo Optimization Algorithm (COA)
[20]	Fruit fly Optimization Algorithm (FOA)
[21]	Artificial Bee Colony (ABC)

Below, five single-objective algorithms and their features are introduced.

3.1.1. Genetic Algorithm (GA)

The main idea of these sort of algorithms is natural selection. Finally, chromosomes (solutions) with beneficial features will either replaced or exist beside previous chromosomes. These beneficial chromosomes are initially generated by performing mutation in one or more genes of existing chromosomes, before being transferred from parents to child(s). The standard genetic algorithm includes four main steps:

- Generation: This step, includes generating an initial population of possible solutions, for example by using a random production method.
- Selection: In this step the previously generated population is assessed by applying a pre-defined fitness function. Afterwards, a subset of it will be selected by one of various available selection techniques, in order to produce a new population within the next steps (crossover and mutation). In other words, selection tries to identify the most appropriate solutions (parents) for generating new solutions (child) that make the next generation. Hence, the selection technique that is applied has an important impact on the algorithm convergence and its speed.
- Crossover: In this step the crossover operator is applied, which produces new chromosomes via combining attributes (genes) of existing chromosomes that has been chosen by the selection technique. There are several crossover methods like uniform, single/multi-point, partially mapped crossover and etc. The purpose of this operator is to generate a child chromosome, that is better than its parents.
- Mutation: In this step the mutation operator is applied, which gives a chance for making random changes in existing chromosomes. It tries to prevent early convergence of the process and to explore the search space with considering genes that did not exist in the first population.

Overall, genetic algorithms are well-known evolutionary algorithms for optimizations problems according to their efficiency for non-convex, non- linear, discrete; and multi-modal issues, which remarkably outperform traditional gradient descent algorithms. However, it should be noted that they cannot guarantee to reach the global optimum even by applying different combinations of hyper-parameters. In addition, these algorithms may require a long time to converge, specially, if the computing sources is insufficient. So, it is significant to have a big-enough population and sufficient number of iterations in order to achieve an acceptable result [22]. As the final point about genetic algorithms, the fitness function design is very important, because this function guides the whole optimization process. Therefore, if it designed inaccurately, infeasible and/or inefficient solutions could be generated during the process.

3.1.2. Genetic Programming (GP)

This algorithm is very related to GAs. Actually, this algorithm uses the same principles in order to program generation via programs which have been modeled as a tree of operands and operators that can be mutated and recombined for optimization. In general, all single-objective GAs pursues an identical main approach. However, they differ in their approach to the previously mentioned primary steps of generation, selection, crossover and mutation.

In most applications, the generation process is completely random, although it can be guided or initialized by an external knowledge, for example knowledge from previous executions of the algorithm. This can help to continue a previous optimization execution, or to optimize a more general approach partially, in order to be applied to reach a new result with possible qualified solutions [23]. Besides, it may bias the process to local optimum and prevent a wider search of the search space. To avoid this possible problem, the initial population can be guided to some extent.

Several recent implementations of these algorithms apply tournament selection technique, because it is scaled constantly and basically elitist, that has been proven to increase the efficiency of the whole process [23].

3.1.3. Particle Swarm Optimization (PSO)

The main idea of this algorithm comes from the social behavior of birds, that give them the important capability of discovering an appropriate place to land [24]. Here, like other introduced algorithms just the main version of PSO is discussed, because it is the fundamental version and for understanding new derivations of PSO, everybody should study the primary version first. This metaheuristic method is inspired by the swarm intelligence concept, which can solve complicated mathematics issues of engineering applications [24, 25].

For clear understanding of the PSO algorithm, an explanation about a flock (swarm) behavior can be helpful for clarification. When a flock of birds are flying above a place, they must choose a location for landing. Therefore, distinguishing the location (the latitude and the longitude) that is the most suitable one for landing the flock is a complicated issue, because it has several parameters. For example, in this decision they should choose a location which minimizes the risk of predators, maximizes the food availability and maximizes the survival possibilities of its members in dangerous conditions.

In this regard, the flock movement could be considered as a harmonious dance; all birds fly coordinated and simultaneously for a period, till the most suitable location for landing is determined and the whole flock lands at the same time. With these features, this issue can be considered as an optimization problem.

With this explanation, the flock movement just occurs if all the flock members have the capability of sharing information among the themselves; otherwise, it is completely possible that each bird land at a different location asynchronously. Thus, with this capability, all flock members would search to find a suitable location for landing and they will detect the best location even if it is discovered by just one of them. In this context, each bird of the flock balances the flock knowledge (which is called global or social knowledge) and its individual knowledge (which is called local knowledge) experience. Each of the flock member flies and searches for the location may consider some criteria to evaluate a location quality for landing, at the same time. So, all of them has the privilege to recognize the most suitable detected location, until it is known by the whole flock.

Applying PSO in different applications has some remarkable advantages in comparison with other optimization methods. For instance, it has simple implementation, fewer parameters, and tuning these parameters are widely discussed in the past research [26].

However, it suffers from few disadvantages. For example: speed of each member (particle) reduces by increasing the iteration number; and converging to the best result(s). In addition, PSO is generally used for issues related to binary decisions. There are many upgraded versions of PSO available that each of them tried to improve it in a way. As an important example, to accelerate the PSO convergence, a simplified version of it, which is called Accelerated Particle Swarm Optimization (APSO), has been proposed and used in some research [27]. In APSO, just the best global location (social knowledge) is calculated by each swarm member.

3.1.4. Ant Colony Optimization (ACO)

The idea of this algorithm inspired by the real-world ant colony, that dispatch the worker ants to find food randomly. These ants leave traces of pheromone (a colored liquid that have smell) behind them. When they find food, these ants will return to their nest and in the path, they will reinforce the trail of pheromone.

Initially, if they face more than one path to choose when they search for food, they will select one of them randomly, because there is not any pheromone trail on these paths. Although, the shorter paths will be more reinforced by ants over time. In this regard, all ants would be attracted to pursue paths that have more pheromone trail; therefore, the whole colony will quickly discover food sources, and most of worker ants will follow the most efficient paths. It should be noted that in most of evolutionary algorithms like ACO, a degree a randomness always exists in order to keep the chance to explore the search space.

In each iteration, ACO algorithm works with many artificial ants that trace a path across the solution space that consists of edges and nodes as solutions, towards the problem objective. Each step in a path is picked by artificial ants using a stochastic approach, which is biased to the pheromone variable existed in each edge and when one node is visited by an artificial ant, it cannot be revisited until the end of the process [22].

ACO algorithm is newer than previously introduced algorithms in this book chapter, but it has been used in various applications. The essence of ACO has the potential to be combined with other available search methods that are suitable for the considered issue, besides, there are many methods which can manage the value of its hyper-parameters in different iterations [28].

ACO has almost the same problems as genetic algorithms have. For example, in cases with adequate computing time and power, it has good effectiveness in solving the given issue. Additionally, this algorithm requires enough number of iterations to be converged and completed. As the last point about this algorithm, it can be highly complicated to implement and run, particularly if the general search approach is a complicate one [22].

3.1.5. Artificial Bee Colony (ABC)

The genuine idea of this method comes from the attitude of bees in a hive, which communicating with each other and searching for food sources. Possible solutions to the considered optimization issue are food sources and the solution quality is measured by the nectar quantity.

Artificial bees in this algorithm are categorized into three groups:
- scouts
- employees
- onlookers

Scouts discover new sources of food and employees assess the available food sources quality. Onlookers obtain the information related to the quality of a checked food source from employees, after that, they assess its quality comparing neighbor sources. Therefore, current solutions could be scrapped and substituted with new ones discovered by scouts.

In general, steps of ABC algorithm are:
- A random solution (food source) is assigned to each employee;
- All employees assess and modify their food source. In this regard, when the new source is better than the past one, the employee withdraws the past and keep the new one;
- Employees share their information with onlookers.
- An onlookers select a food source according to fitness that has been calculated using all employees' information and make a modification. If this modification is better, it will be replaced by the original source and scouts will replace discarded food sources with new ones.

Besides the first version of ABC algorithm, many derivations of it with similar concepts have been proposed recently. For instance, one version of it pursues almost the same flow, however, it uses another approach for neighborhood search, and another simulation mechanism for inter-bee communication [29].

Another recent example defined an upgraded version which applies the concept of swap-sequences to achieve better results in issues that have the travelling salesman model. This version could attain almost the same level of performance as other common methods for solving similar issues in this area [30].

The most significant problem with this algorithm is the lack of practical experiments that can be applied on real-world issues, which show promising results or performance progression in comparison with other related methods. However, ABC algorithm has been used in some domains, its applications were not as wide as other algorithms [31]. Thus, researchers should spend some time to get familiar with its features related to the given application, although, it may also increase the chance of achieving remarkable results specially if it has not been used for that application before. Besides, similar to many swarm intelligence methods, process speed could be counted as a limitation for ABC algorithm [22].

In Table 2, some of the main single-objective evolutionary algorithms are compared with each other [32].

Table 2. Qualitative comparison of the most used evolutionary algorithms [32]

	GA	GP	PSO	ACO	ABC
Implementation	Med	Med	Easy	Med	Hard
Require ranking of solutions	Yes	Yes	No	No	Yes
Population size effect on execution time	Expo	Expo	Lin	Lin	Lin
The best solution effect on population	Med	Med	Most	Most	Most
Average fitness could not get worse	False	False	False	False	False
Possibility of premature convergence	Med	Med	High	High	High
Density (continuity) of search space	Less	Less	More	More	More
Find suitable solutions without local search	Less	Less	More	More	More
Homogeneous subgrouping improves convergence	Yes	Yes	Yes	Yes	-

In Table 2, Expo is the short form of Exponential, Med is the short form of Medium and Lin is the short form of Linear.

3.2. Multi-Objective and Many-Objective Algorithms

As it can be recognized from the name of these algorithms, they represent the idea of having multiple objectives in one problem, not a single objective. The behavior of these objectives could be against each other (conflict) and complicated. A common example in different industrial applications is detecting the best trade-off place among power consumption, cost, time and performance. These objectives could be met to various degrees, specially by allowing the end users to choose their priorities. For example, too much performance could increase the power consumption and costs a lot.

In contrast, low cost could reduce the performance and power consumption. Evolutionary algorithms are highly suitable for multi-objective problems, because their fitness functions are about a compromise along the Pareto-fronts and dominants [7].

Multi-Objective evolutionary algorithms are very appropriate for solving problems that have several complicated objectives. These problems usually called Multi-Objective Problems (MOPs). In these problems, conflicts between two or more objectives commonly prevent achieving a single optimal solution. Therefore, a collection of trade-off results, named the Pareto optimal set is returned. In most multi-objective evolutionary algorithms, a solution dominates the other one if none of its objectives is worse and it is strictly superior in at least one objective [40]. Solutions existed in the first Pareto-front are non-dominated by any other one(s) of the solution space. Using this idea, multi-objective evolutionary algorithms have proven to be very appropriate for various complicated MOPs [40].

Generally, a MOP consists n parameters which are decision variables, s objective functions, and m constraints. The last two ones are functions of the parameters. The main purpose of an optimization process is to maximize [40].

$$y = f(x) = (f_1(x), f_2(x), \ldots, f_s(x)) \quad (1)$$

where,

$$c(x) = (c_1(x), c_2(x), \ldots, c_m(x)) \leq 0$$

$$x = (x_1, x_2, \ldots, x_n) \in X$$

$$y = (y_1, y_2, \ldots, y_s) \in Y$$

In the above-mentioned equations, X is the decision space and Y is the objective space. Additionally, x and y are the decision and the objective vectors, respectively. Furthermore, the constraint $c(x) \leq 0$ specifies the collection of possible solutions [40].

It is very important to note that in some applications, although there was more than one objective, using the weighted summation of these objectives, a single-objective algorithms have been applied to solve them. However, it is obvious that the use of multi-objective algorithms that are designed specifically for these issues can lead to better results.

Commonly, MOPs with four or more objectives are known as many-objective problems (MaOPs), however, some research defined problems with at least three objectives as MaOPs [33]. In the following, one well-known multi-objective and one many-objective algorithm and their features are introduced.

3.2.1. Non-Dominated Sorting Genetic Algorithm II (NGSA-II)

NSGA is one of the first multi-objective algorithms. It produces one approximate Pareto-front according to a population of candidate solutions in a given optimization issue, that is assessed comparing a number of objectives (from 1 to n), that are typically specified for minimization or maximization [34]. Each Pareto-front contains a collection of solutions, that the performance of each of them can be increased only in one objective via decreasing it in another objective. So, the solutions of this Pareto-front dominate all other existing solutions of the search space.

The most used version of this algorithm called NSGA-II attains the improvements of an approximate Pareto-front using an iterative manner [34]. It is a computationally fast algorithm that have the non-dominated sorting approach. In the first stage, a population with a certain size is formed. It can be created randomly or using some other methods for creating the initial population with external knowledge. Thereupon, for each solution of the population, two values would be calculated:

- The number of solutions that dominate the considered solution (domination count).
- The collection of all solutions which the considered solution dominates (dominating count).

In this regard, the domination count for all solutions within the first non-dominated rank would be zero. Then, for each of them, all members of the dominated collection are met, and their dominating count decreased by one and if the dominating count of a collection is decreased to zero, it is attached to the second non-dominated level.

This process repeated in the same way till all solutions of the population would be added to a certain rank. Afterwards, for each solution a fitness function is calculated which is equal to its rank (considering 1 as the best and rest of them in descending order).

In the next stage, the chosen selection technique and the two pre-defined operators could be applied to generate a child population with the size of n. Then this generated population is combined with the last step population, which creates a major population with the size of $2n$. Next, this population will be ordered via the non-dominating ranks similar to the previous description.

For all individual solutions the value of crowding distance parameter should be calculated. This value is calculated by a function that measures the distance from the nearest other solutions in the same rank (Pareto-front). The larger the measured size, the better, as it means that the degree of variety is high between the solutions of the same rank. The crowding distance of outlier solutions is always equal to infinity. By this means, members within their ranks are sorted using their crowding distance value. Therefore, to extract a population with the size n from the major population, the first n members of the sorted population are chosen to create a new population. These chosen solutions have the best ranks in the entire solution space, further, they are the most diverse and varied individuals of a final rank to form this new population with exact size of n.

Once these steps finished, the chosen selection technique is again applied. This time it will also use the crowding comparison value, in order to consider both crowding distance and rank. This cause the parents of a new population with the size of n to be selected. As a result, the iteration could be continued by merging the populations, scoring and ranking, then extracting n members and iterating again till a termination condition is visited [34].

Despite numerous advantages of NSGA-II, it has some weaknesses. However, these weaknesses are almost the same amongst many multi-objective optimization methods.

Considering the nature of NSGA-II, evaluations of an objective function should be done many times. As the approach is often used for complicated real-world issued that have some objective functions, the entire process could require extremely long runtimes. For solving this problem, some algorithms are proposed recently that applied the main idea of NSGA-II. Several changes and improvements in these algorithms tried to apply metaheuristics or other methods to decrease the number of times that an objective function is evaluated.

Similar to the standard GA, potentially NSGA-II can be parallelized too. Thus, the other idea which has been studied is parallel execution of objective function evaluations among various devices.

Another problem happens when more knowledge about the data and the solution space gained, so a more guided mutation procedure can result in a remarkable increase of the convergence speed. Predictably, it is risky because this can result in missing the gain that during the process solutions which could be unclear for analysis, be identified [22]. The next weakness is that a NSGA-II method generates a Pareto-front. It includes some solutions with different scores in various objectives, which dominate all other individual solutions. As mentioned before, the final solution should to be chosen from this Pareto-front to be optimized, although, it may increase the complexity of the process. However, it allows for objectives to be prioritized.

All in all, NSGA-II has several benefits which make it appropriate for different types of optimization issues like low computational complexity, capturing multiple solutions in a Pareto-front, addressing non-linear and dis-continuing issues. Among previous related works that applied multi-objective algorithms, NSGA-II was the most popular and the most used one. This is evidence of its sufficiency in solving optimization issues in various applications [35].

3.2.2. Non-Dominated Sorting Genetic Algorithm III (NSGA-III)

The general behavior of NSGA-III is almost the same as the previously mentioned algorithm (NSGA-II). It utilizes the similar approach in order to pursue some of the evolutionary steps. Accordingly, it uses similar approach applied by NSGA-II to generate the initial solution population [36]. Afterwards, it utilizes similar operators (mutation and crossover) applied by NSGA-II, in order to produce a child population from parents.

Nonetheless, there are some differences between version III and II of NSGA. For example, NSGA-III uses another selection technique to produce a new population for the next generation, which merges current and child population [37].

In addition, the NSGA-III process begins, like NSGA-II, by categorizing different solutions in the merged population considering their non-domination levels (Pareto-fronts) and next choosing the Pareto-fronts once at a time to produce the next population. The process continues till the population size becomes equal or greater than the initial solutions. Once the new population size is more than the needed size, the last processed Pareto-front cannot be completely transferred to the new population. Thus, NSGA-III applied a different selection technique in order to determine which solutions from this front should be chosen for the new population. This selection technique uses a set of reference points that are distributed on the inherently normalized hyper-plane widely and uniformly, to the objectives of the given optimization issue. In this regard, the algorithm tries to select solutions from the first front which have connections to each of these reference points. Therefore, the algorithm elevates selecting the well-distributed and diverse non-dominated solutions, with the intention to maintain the distribution and diversity in the new generated population.

The termination condition in both versions of NSGA is the same. After visiting this condition and finishing the algorithm execution, NSGA-III returns the Pareto members of the last generation population as the final output.

Overall, in both of these algorithms each solution is equivalent to a possible scaling plan and they both begin from a random initial population. In every generation, they apply similar operators to produce a child population from the current one. However, they differ in terms of the selection technique that they apply to decide which solutions from these two populations (current and child population) are suitable to be transferred to the next generation population.

As mentioned before, the NSGA-II selection technique first puts the non-dominated solutions in the merged population and after that calculates crowding distance parameter for them. This parameter represents the solution distance to its neighbor solutions. Afterwards, the algorithm tries to select non-dominated solutions with greater crowding distances. Hence, the algorithm elevates selecting diverse non-dominated solutions. Although, this approach could not guarantee to select the most suitable distributed non-dominated solutions.

In contrast, the selection technique of NSGA-III first puts the non-domination solutions in the merged population and after that it considers the solutions connection to the reference points. Hence, the algorithm applies a set of well-distributed reference points. Afterwards, the algorithm tries to select non-dominated solutions that are connected with each of these reference points. Therefore, this algorithm elevates selecting well-distributed and diverse non-dominated solutions, with the intention to maintain the distribution and diversity of the next generated population [37].

Applying this selection technique with well-spread reference points, helps NSGA-III to increase the probability of achieving more qualified Pareto sets considering both distribution and diversity of the non-dominated solutions. In Table 3, Some of the most used multi-objective and many-objective evolutionary algorithms are mentioned.

Table 3. Multi/Many-objective evolutionary algorithms

Reference	Algorithm name
[34]	Non-dominated Sorting Genetic Algorithm II (NSGA-II)
[38]	Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D)
[39]	Pareto Envelop-based Selection Algorithm (PESA)
[40]	Multi-objective selection based on dominated hyper-volume (SMS-EMOA)
[41]	Non-dominated Sorting Genetic Algorithm III (NSGA-III)
[42]	Many-Objective Directed Evolutionary Line Search (MODELS)
[43]	Multi-Objective Particle Swarm Optimizer (MOPSO)
[16]	Multi-objective Flower Pollination Algorithm (MOFPA)

4. APPLICATIONS

Evolutionary algorithms have been applied in applications that classical or new stochastic methods were unsuccessful in finding solutions or they had unacceptable time and/or computational complexity [44, 45]. Usually, this happens because of several dimensions, resources constraints or complicated functionality.

By increasing available resources of the application, these problems can be solved. Actually, by considering infinite computation capability and resources, a classical exploitative or stochastic method can achieve an acceptable result.

Nonetheless, evolutionary algorithms can be discussed as the last solutions for the mentioned problems [7]. These issues have complicated nature and extreme domain size. In addition, their various objectives eliminate the possibly to explore the while search space. Therefore, in issues with these specifications, good-enough solutions are also acceptable, as detecting the proven global optima is not possible (and not necessary). Thus, evolutionary algorithms commonly are not suitable for small and simple issues, because classical methods can easily detect the most suitable solution(s) for these issues.

Evolutionary algorithms have been used in various application types, from conceptual design creation to variable optimization issues. These problem types could be divided into variable optimization, new structural design and improvement categories [45]:

1) Variable optimization means searching the given problem variable space for a suitable solution. In some cases, there are large number of variables in the space which should be searched. Therefore, in these issues the search needed to be on a larger scale in comparison with a classical programming issue. This point is exactly where the goal can be precisely specified.

2) New structural design includes creating a quite new solution such as a mechanical design or a program.

3) Improvement is equivalent to a situation that an existing solution is entered and evolutionary algorithms try to discover better solutions for it.

Focusing more on evolutionary algorithms applications, undoubtedly, they have been used in various issues from industrial and commercial applications to leading edge research in different fields like natural language processing and planning future city landscapes for town designers [46], [47]. In the following some of the main applications of the previously introduced evolutionary algorithms are mentioned. However, obviously it is impossible to explain all of their applications here, so only a few examples are reflected within this chapter.

GAs have been used in numerous research and real-world issues, over a wide range of fields such as classic optimization issues, engineering applications, protein folding, industrial problems and commercial issues [48]. According to the parallel nature of these algorithms, they suggest several various possibilities for acceleration over various machines and cores, besides taking advantage of having several compute powers available in different places, as has been proven in some recent research [49]. In addition, GAs is almost able to utilize the parallelization abilities of Graphics Processing Units (GPU) [50].

As well as GAs, PSO algorithms can be used in various types of issues in the most diverse scientific fields. For instance, they have been applied in healthcare applications for diagnosing the type of leukemia via microscopic imaging. Also, in problem related to economics, they have been applied to analyze restricted and unrestricted risk investment portfolios to reach the optimal risk portfolios [51]. Another example would be its application in natural language processing task, which results in high accuracy of outputs in question answering systems [52].

In the literature, optimization methods like PSO algorithm are applied with the intention to enhance the systems heat transfer or even in cases in order to calculate the heat transfer ratio. In thermodynamics, there are some researches which works on optimizing the performance of thermal systems like hybrid diesel-ORC/photovoltaic system, diesel engine-organic Rankine cycle and Integrated Solar Combined Cycle (ISCC) power plants [51].

PSO algorithm has been applied on geometric optimization problems too, with the aim of discovering the optimal configurations for the system which fits with constraints of the design perfectly. In this regard, there are some researches including geometric optimization of radiative enclosures and optical-geometric optimization of solar concentrators which satisfy heat flow and distribution of temperature [51].

Similarly, ACO has been used in various significant fields. This algorithm was firstly applied on the travelling salesman issue [51]. This issue is known as NP-hard and very challenging problem, according to the number of cities that taken part. Then, ACO has been utilized to different engineering issues like machine learning optimization, vehicle routing issues and bio-informatics issues, which could achieve remarkable results.

It has been proven that ABC algorithm have acceptable performance on artificial experiments in comparison with other common methods. As an example, for travelling salesman issue with some modifications it could achieve notable results. Besides, they have been used with remarkable outcomes to the route optimization issue for network communications, and also to color image segmentation [51].

Likewise, NSGA-II has been used widely over industry, especially in the engineering problems that there are many complex multi-objective problems for which this algorithm can be applied on [53, 54]. Further metaheuristic methods have been used in water distribution systems to detect the best placement for pressure sensors, by applying an altered version of NSGA-II algorithm and applying artificial neural networks (ANN) in order to estimate final results, instead of assigning solutions to bad and good classes [55].

It should worth noting that the real performance of evolutionary algorithms is not just in industrial problems. Their application in artificial intelligence systems, the first time resulted in introducing a new theory called planar regeneration mechanism [56].

All in all, there are plenty of research that investigate evolutionary algorithms and their applications in industrial problems, although very rarely these studies work on applications which have been exploited in real-world issues. Therefore, it can be concluded that the theory does not support the practice and a huge gap exists between them. Theoretical consequences on attributes like exploration and exploitation, convergence and diversity are not beneficial enough for real-world issues. Important subjects in practical view are noise and constraint handling, robustness, and also multi-objective optimization, where in some improvements are also reported, but in most cases, they are examined on naïve issues. Thus, their feasibility for evolutionary-based practical applications is not analyzed adequately.

The industrial issues are extremely complicated. Thus, designing a suitable mathematical schema, which for evolutionary algorithms is equivalent to suitable objective function, considering an industry application would be very necessary. In addition, the quality of the selected objective function would have a notable effect on the final results captured by evolutionary algorithms. Another issue that should be considered is repeatability of these algorithms. Since evolutionary algorithms are stochastic methods, each time the chosen algorithm is performed, a different result could be captured. So, much attention needs to be paid on repeatability feature of the results produced by evolutionary algorithms, specifically for their application in industry.

4.1. Metaheuristic Methods for Power Energy Systems

In power and energy systems, as an important example of engineering applications, metaheuristic methods have been used widely, to address various problems related to [57-59]:

1. Planning
2. Operation
3. Control
4. Prediction
5. Security
6. Reliability
7. Demand management

In the case of information coding, the most prosperous implementations of GAs would not apply binary string encoding, though, it applies representations which are fitted to the considered application, and its operators (mutation and crossover) are re-defined accordingly [60]. Although, binary coding models are still common in several issues of power and energy systems. Alternatively, evolutionary algorithms where in the binary values are substituted with real numbers or integer, are proper for issues with particular features.

Table 4 represents the most popular metaheuristic methods that have been applied on common problems in this area and could address them efficiently [59]. From this table, it can be recognized that GA is the most popular algorithm for almost all problems in this field. Perhaps the reason is its longer history in comparison with other algorithms. Besides, these algorithms placed in next rankings, respectively: PSO, SA (Simulated Annealing), TS (Tabu Search) and DE (Differential Evolution) [61-63].

Table 4. Most popular metaheuristics in power and energy systems sphere [59, 62, 63]

Power and Energy Systems issue	Most popular Metaheuristic methods
Unit commitment (UC)	GAs, PSO
Economic Dispatch (ED)	GAs, PSO, DE
Optimal Power Flow (OPF)	GAs, PSO, SA, DE
Distribution System Reconfiguration (DSR)	GAs, PSO, ACO, SA
Maintenance Scheduling (MS)	GAs, PSO, SA, TS
Transmission Network Expansion Planning (TNEP)	GAs, PSO, SA, TS
Load and Generation Forecasting (LGF)	GAs, PSO, SA
Distribution System Planning (DSP)	GAs, PSO, TS

Some main examples are pointed out in Table 4, in order to represent how metaheuristic methods can be an appropriate and a more successful choice for mathematical programming implements for solving a large-scale optimization issue in the field of power and energy systems. A prevalent observation is that the problem size has an important role. When the problem size is not very large, that mathematical programming tools or exhaustive search could be applicable, then using metaheuristics is not reasonable, because these methods can find exact solutions with acceptable computational cost. There is just one exception, which is the case that the goal is to test a metaheuristic method on an issue which its global optimum is known, to evaluate its efficiency in detecting the global optimum, before using it for real-world large-scale issues. Therefore, the large-scale parameter setting and implementation will probably be useless if the method cannot detect the optimal solution on a small-size issue after sufficient test executions [62].

4.2. The Most Suitable Evolutionary Algorithm for a Given Application

All introduced approaches in this book chapter and many others that have not been mentioned here are from the same category: evolutionary algorithms for optimization issues [46, 65]. The primary question here is: Which of them is the most appropriate one for a considered area? In general, by extending the answer to all metaheuristics, not only evolutionary algorithms, it seems that the best answer is simple: the known one is the best. In this way, it opens an excellent opportunity to describe subject perfectly in terms of requirements of this algorithm and awareness of the sensitivity degree of this algorithm to different parameters which makes possible to manage the process precisely. For instance, several evolutionary algorithms are used in power and energy supply systems applications and it is not possible to consider just one of these algorithms as the best one for this specific area. No doubt that the same approach can be pointed out for other industrial spheres as well.

There are available numerous kinds of evolutionary algorithms [65]. Selecting the optimal algorithm for a particular application depends on its features like:

1. Accuracy
2. Speed of processing
3. Data type
4. Amount of required/needed data
5. Implementation simplicity
6. Any others

Generally, the above-mentioned variations could be used to shortlist some algorithms, while it is very complicated to select exactly one of evolutionary algorithms in the beginning stage of application that will be able to achieve expected outcomes. In this regard, trial and error besides repetition could usually be useful. Therefore, it might be effective to identify the evolutionary algorithms as potential suitable approaches, by throwing the data into them, executing in serial or parallel manner, and finally evaluating their performance in order to be able to choose the best algorithm(s) for the given problem solution.

5. CONCLUSION AND FUTURE WORKS

In recent decades, metaheuristic methods introduced a combined research area. The research topics that apply and study the capability of these methods are continuously evolving, particularly because of the emergence of novel theoretical advancements to supply fundamentals for techniques, the compilation of many technological improvements, and the formulation of new algorithmic suggestions which allow metaheuristic methods to comply the requirements that scientists find in their research. The great usefulness of these methods in various applications attracts many researchers to work on their potentials and capabilities. Therefore, in various domains like engineering, their applications are very diverse.

This book chapter presented a review about evolutionary algorithms and their different types, as a main sub-field of metaheuristic methods and their advances in various research areas, including some significant comments about their applications in real-world problems, for instance, power energy systems. In other words, it reviews one of the most popular and important methods, which is being used a lot for solving optimization problems and practical issues, in a various field. Besides, some points were mentioned about the operation, advantages and disadvantages of the most important examples of these algorithms.

As future works in evolutionary algorithms, some primary directions can be mentioned. Undoubtedly, the first one is working on hybridization of two or more algorithms to achieve more accurate results. Recently, an increasing number of researches have studied hybrid algorithms. In addition, several papers have investigated different modifications of current evolutionary algorithms to progress their efficiency.

There is no doubt that, current evolutionary algorithms have some problems, particularly in industrial applications, that can be solved in future works. Sometimes in real-world problems, a high-dimensional function should be optimized, so this process would be complicated and requires lots of computational time. In this regard, the chosen evolutionary algorithm should be designed with an easy implementation in order to perform in parallel. Thus, the computational time would be reduced. A greater future effort in this issue is needed, as this can be a vital specification to decide whether a possible method is beneficial in practice or not.

According to many studies, in the future new evolutionary algorithms will be introduced, therefore, the research issues related to them will always remain a trend topic for scientists. Thus, by writing this book chapter, the attempt was made to provide a comprehensive overview of metaheuristic methods with focus on evolutionary algorithms to pave the way for those who are desiring to use pointed out methods/instruments in research activities.

REFERENCES

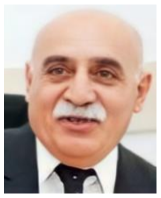
[1] S. Nesmachnow, "An Overview of Metaheuristics: Accurate and Efficient Methods for Optimisation", International Journal of Metaheuristics, Vol. 3, No. 4, pp. 320-347, 2014.
[2] M. Solgi, O. Bozorg Haddad, H. Loaiciga, "Meta-Heuristic and Evolutionary Algorithms for Engineering Optimization", John Wiley and Sons, pp. 280-292, 2017.
[3] S. de Leon Aldaco, H. Calleja, J. Aguayo, "Metaheuristic Optimization Methods Applied to Power Converters: A Review", IEEE Transactions on Power Electronics, Vol. 30, pp. 1-16, 2015.
[4] H. Stegherr, M. Heider, J. Hahner, "Classifying Metaheuristics: Towards a Unified Multi-Level Classification system", Natural Computing, 2020.
[5] C. Blum, A. Roli, "Hybrid Metaheuristics: An Introduction", Hybrid Metaheuristics, pp. 1-30, Berlin, Heidelberg, 2008.
[6] M.J. Reddy, D.N. Kumar, "Evolutionary Algorithms, Swarm Intelligence Methods, and Their Applications in Water Resources Engineering: A State-of-the-Art Review", H2Open Journal, Vol. 3, No. 1, pp. 135-188, 2020.
[7] A.N. Sloss, S. Gustafson, "2019 Evolutionary Algorithms Review", Genetic and Evolutionary Computation, pp. 307-344, Cham, Switzerland, 2020.
[8] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs", Berlin, Heidelberg, Germany, 1992.
[9] J.R. Koza, "Genetic Programming as a Means for Programming Computers by Natural Selection", Statistics and Computing, Vol. 4, pp. 87-112, 1994.
[10] X. Yang, "Firefly Algorithms for Multimodal Optimization", (SAGA) Stochastic Algorithms: Foundations and Applications, pp. 169-178, Berlin, Heidelberg, Germany, 2009.
[11] E. Atashpaz Gargari, C. Lucas, "Imperialist Competitive Algorithm: An Algorithm for Optimization Inspired by Imperialistic Competition", IEEE Congress on Evolutionary Computation, Singapore, 2007.
[12] J. Kennedy, R. Eberhart, "Particle Swarm Optimization", International Conference on Neural Networks (ICNN), Perth, WA, Australia, 1995.

[13] X.S. Yang, "Nature-Inspired Metaheuristic Algorithms", Luniver Press, Bristol, UK, 2008.
[14] P. Moscato, "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts - Towards Memetic Algorithms", California Institute of Technology, Pasadena, 1989.
[15] F. Moyson, B. Manderick, "The Collective Behavior of Ants: An Example of Self-Organization in Massive Parallelism", Artificial Intelligence Laboratory, Vrije Universiteit, Brussel, Belgium, 1988.
[16] X.S. Yang, "Flower Pollination Algorithm for Global Optimization", International Conference on Unconventional Computing and Natural Computation, Berlin, Heidelberg, Germany, 2012.
[17] M.M. Eusuff, K.E. Lansey, "Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm", Journal of Water Resources Planning and Management, Vol. 129, No. 3, pp. 10-25, 2003.
[18] S.C. Chu, P.W. Tsai, J.S. Pan, "Cat Swarm Optimization", Trends in Artificial Intelligence, The 9th Pacific Rim International Conference on Artificial Intelligence, Berlin, Heidelberg, Germany, 2006.
[19] R. Rajabioun, "Cuckoo Optimization Algorithm", Applied Soft Computing, Vol. 11, No. 8, pp. 5508-5518, 2011.
[20] W.T. Pan, "A New Fruit Fly Optimization Algorithm: Taking the Financial Distress Model as an Example", Knowledge-Based Systems, Vol. 26, pp. 69-74, 2012.
[21] D. Karaboga, "An Idea Based on Honey Bee Swarm for Numerical Optimization", Department of Computer Engineering, Engineering Faculty, Erciyes University, Kayseri, Turkey, 2005.
[22] X. Fan, W. Sayers, S. Zhang, Z. Han, L. Ren, H. Chizari, "Review and Classification of Bio-inspired Algorithms and Their Applications", Journal of Bionic Engineering volume, Vol. 17, pp. 611-631, 2020.
[23] P.V. Paul, N. Moganarangan, S.S. Kumar, R. Raju, T. Vengattaraman, P. Dhavachelvan, "Performance Analyses Over Population Seeding Techniques of the Permutation-Coded Genetic Algorithm: An Empirical Study based on Traveling Salesman Problems", Applied Soft Computing, Vol. 32, pp. 383-402, 2015.
[24] B.S. Gomes de Almeida, V.C. Leite, "Particle Swarm Optimization: A Powerful Technique for Solving Engineering Problems", Swarm Intelligence - Recent Advances, New Perspectives and Applications, Intech Open, pp. 1-21, 2019.
[25] L. Hantash, T. Khatib, M. Khammash, "An Improved Particle Swarm Optimization Algorithm for Optimal Allocation of Distributed Generation Units in Radial Power Systems", Applied Computational Intelligence and Soft Computing, Vol. 2020, pp. 1-20, 2020.
[26] H. Masoud, S. Jalili, S.M.H. Hasheminejad, "Dynamic Clustering using Combinatorial Particle Swarm Optimization", Applied Intelligence, Vol. 38, pp. 289-314, 2013.
[27] N. Tohidi, C. Dadkhah, "Improving the Performance of Video Collaborative Filtering Recommender Systems using Optimization Algorithm", International Journal of Nonlinear Analysis and Applications, Vol. 11, No. 1, pp. 283-295, 2020.
[28] B.C. Mohan, R. Baskaran, "A Survey: Ant Colony Optimization Based Recent Research and Implementation on Several Engineering Domain", Expert Systems with Applications, Vol. 39, No. 4, pp. 4618-4627, 2012.
[29] D.T. Pham, M. Castellani, "A Comparative Study of the Bees Algorithm as a Tool for Function Optimisation", Cogent Engineering, Vol. 2, No. 1, pp. 1-28, 2015.
[30] I. Khan, M.K. Maiti, "A Swap Sequence Based Artificial Bee Colony Algorithm for Traveling Salesman Problem", Swarm and Evolutionary Computation, Vol. 44, pp. 428-438, 2019.
[31] J. Ning, C. Zhang, B. Zhang, "A Novel Artificial Bee Colony Algorithm for the QoS Based Multicast Route Optimization Problem", Optik, Vol. 127, No. 5, pp. 2771-2779, 2016.
[32] V. Kachitvichyanukul, "Comparison of Three Evolutionary Algorithms: GA, PSO, and DE", Industrial Engineering and Management Systems, Vol. 11, No. 3, pp. 215-223, 2012.
[33] B. Li, J. Li, K. Tang, X. Yao, "Many-Objective Evolutionary Algorithms", ACM Computing Surveys, Vol. 48, No. 1, pp. 1-35, 2015.
[34] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, "A Fast and Elitist Multi Objective Genetic Algorithm: NSGA-II", IEEE Transactions on Evolutionary Computation, Vol. 6, No. 2, pp. 182-197, 2002.
[35] I. Alothaimeen, D. Arditi, "Overview of Multi-Objective Optimization Approaches in Construction Project Management", Multicriteria Optimization - Pareto-Optimality and Threshold-Optimality, IntechOpen, London, UK, 2019.
[36] H. Seada, K. Deb, "Effect of Selection Operator on NSGA-III in Single, Multi, and Many-Objective Optimization", IEEE Congress on Evolutionary Computation (CEC), Sendai, Japan, 2015.
[37] V. Yannibelli, E. Pacini, D. Monge, C. Mateos, G. Rodriguez, "A Comparative Analysis of NSGA-II and NSGA-III for Autoscaling Parameter Sweep Experiments in the Cloud", Scientific Programming, Vol. 2020, pp. 1-17, 2020.
[38] Q. Zhang, H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition", IEEE Transactions on Evolutionary Computation, Vol. 11, No. 6, pp. 712-731, 2008.
[39] D. Corne, J.D. Knowles, M.J. Oates, "The Pareto Envelop-Based Selection Algorithm for Multi-Objective Optimization", International Conference on Parallel Problem Solving from Nature, Berlin, Heidelberg, Germany, 2000.
[40] N.B. Hochstrate, B. Naujoks, M. Emmerich, "SMS-EMOA: Multiobjective Selection Based on Dominated

Hypervolume", European Journal of Operational Research, Vol. 181, No. 3, pp. 1653-1669, 2007.
[41] H. Jain, K. Deb, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems with Box Constraints", IEEE Transactions on Evolutionary Computation, Vol. 18, No. 4, pp. 577-601, 2014.
[42] E.J. Hughes, "Many-Objective Directed Evolutionary Line Search", The 13th Annual Conference on Genetic and Evolutionary Computation, 2011.
[43] C.A. Coello, M.S. Lechuga, "MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization", The 2002 Congress on Evolutionary Computation, Honolulu, HI, USA, 2002.
[44] N. Tohidi, R.B. Rustamov, "Geographic Information Systems in Geospatial Intelligence", A Review of the Machine Learning in GIS for Megacities Application, Intech Open, October 2020.
[45] D. Greiner, J. Periaux, D. Quagliarella, J. Magalhaes-Mendes, B. Galvan, "Evolutionary Algorithms and Metaheuristics: Applications in Engineering Design and Optimization", Mathematical Problems in Engineering, Vol. 2018, pp. 1-4, 2018.
[46] N. Tohidi, C. Dadkhah, R.B. Rustamov, "Optimizing the Performance of Persian Multi-Objective Question Answering System", The 16th International Conference on Technical and Physical Problems of Electrical Engineering (ICTPE), pp. 110-116, Istanbul, Turkey, 12-13 October 2020.
[47] W. Yao, Y. Ding, "Smart City Landscape Design Based on Improved Particle Swarm Optimization Algorithm", Complexity, Vol. 2020, p. 10, 2020.
[48] M.L. Islam, S. Shatabda, M.A. Rashid, M.G.M. Khan, M.S. Rahman, "Protein Structure Prediction from Inaccurate and Sparse NMR data Using an Enhanced Genetic Algorithm", Computational Biology and Chemistry, Vol. 79, pp. 6-15, 2019.
[49] A.S. Akopov, L.A. Beklaryan, M. Thakur, B.D. Verma, "Parallel Multi-Agent Real-Coded Genetic Algorithm for Large-Scale Black-Box Single-Objective Optimisation", Knowledge-Based Systems, Vol. 174, pp. 103-122, 2019.
[50] J. Luo, S. Fujimura, D. El Baz, B. Plazolles, "GPU Based Parallel Genetic Algorithm for Solving an Energy Efficient Dynamic Flexible Flow Shop Scheduling Problem", Journal of Parallel and Distributed Computing, Vol. 133, pp. 244-257, 2019.
[51] B.S.G. de Almeida, V.C. Leite, "Particle Swarm Optimization: A Powerful Technique for Solving Engineering Problems", Swarm Intelligence - Recent Advances, New Perspectives and Applications, IntechOpen, London, UK, 2019.
[52] N. Tohidi, S.M.H. Hasheminejad, "Optimizing Question Answering Systems by Accelerated Particle Swarm Optimization (APSO)", Signal and Data Processing, Vol. 20, No. 1, 2022.
[53] W. Sayers, D. Savic, Z. Kapelan, "Performance of LEMMO with Artificial Neural Networks for Water Systems Optimization", Urban Water Journal, Vol. 16, No. 1, pp. 21-32, 2019.
[54] N. Tohidi, S.M.H. Hasheminejad, "MOQAS: Multi-Objective Question Answering System", Journal of Intelligent & Fuzzy Systems, Vol. 36, No. 4, pp. 3495-3512, 2019.
[55] Q. Wang, L. Wang, W. Huang, W. Zhihong, S. Liu, D.A. Savic, "Parameterization of NSGA-II for the Optimal Design of Water Distribution Systems", Water, Vol. 11, No. 5, 2019.
[56] M. Ivankovic, R. Haneckova, A. Thommen, M.A. Grohme, M. Vila Farre, S. Werner, J.C. Rink, "Model Systems for Regeneration: Planarians", Development, Vol. 146, p. 12, 2019.
[57] K.Y. Lee, M.A. El Sharkawi, "Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems", Wiley-IEEE Press, p. 624, Hoboken, NJ, USA, 2008.
[58] K.Y. Lee, Z.A. Vale, "Applications of Modern Heuristic Optimization Methods in Power and Energy Systems", Wiley-IEEE Press, p. 896, Hoboken, NJ, USA, 2020.
[59] G. Chicco, A. Mazza, "Heuristic Optimization of Electrical Energy Systems: Refined Metrics to Compare the Solutions", Sustainable Energy, Grids and Networks, Vol. 17, No. 100197, 2019.
[60] E.D. Taillard, L. Mgambardella, M. Gendreau, J.Y. Potvin, "Adaptive Memory Programming: A Unified View of Metaheuristics", European Journal of Operational Research, Vol. 135, No. 1, pp. 1-16, 2001.
[61] Y. del Valle, G.K. Venayagamoorthy, S. Mohagheghi, J.C. Hernandez, R.G. Harley, "Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems", IEEE Transactions on Evolutionary Computation, Vol. 12, No. 2, pp. 171-195, 2008.
[62] G. Chicco, A. Mazza, "Metaheuristic Optimization of Power and Energy Systems: Underlying Principles and Main Issues of the Rush to Heuristics", Energies, Vol. 13, No. 19, pp. 1-38, 2020.
[63] H. Xiao, Z. Dong, L. Kong, W. Pei, Z. Zhao, "Optimal Power Flow Using a Novel Metamodel Based Global Optimization Method", Energy Procedia, Vol. 145, pp. 301-306, 2018.
[64] N. Tohidi, C. Dadkhah, R.B. Rustamov, "Optimizing Persian Multi-Objective Question Answering System", International Journal on Technical and Physical Problems of Engineering (IJTPE), Issue 46, Vol. 13, No. 1, pp. 62-69, March 2021.
[65] B. Baydar, H. Gozde, M. Ari, M.C. Taplamacioglu, "A Research on Evolutionary Computation Techniques in Optimal Power Flow Solution", International Journal on Technical and Physical Problems of Engineering (IJTPE), Issue 33, Vol. 9, No. 4, pp. 26-33, December 2017.

BIOGRAPHIES

Nasim Tohidi was born in Tehran, Iran, on March 27, 1993. She received the B.Sc. and M.Sc. degree in software engineering from Alzahra University, Tehran, Iran. Now, she is a Ph.D. student in computer engineering (artificial intelligence) at K. N. Toosi University of Technology, Tehran, Iran. She awarded honorary admission to the graduate program and also received some scholarships for having high academic records and achievements. Her research interests include natural language processing and evolutionary algorithms.



Rustam B. Rustamov was born in Ali Bayramli, Azerbaijan, on May 25, 1955. He has graduated Ph.D. at the Russian Physical-Technical Institute, S. Petersburg, Russia. He was invited for the work at the European Space Agency within the Framework of the United Nations Program on Space Applications at the European Space Research and Technology Center, The Netherlands. He has appointed for the United Nations Office for Outer Space Affairs Action Teams (member, Vienna, Austria), United Nations Economical and Social Commission for Asia and the Pacific (national focal point, Thailand), International Astronautically Federation (Federation's contact, France) as well as Co-Chair of the International Astrobaitical Congress IAC2022. Currently, he works for Azercosmos OJSC, Baku, Azerbaijan as an advisor on space science and technology. He has mainly specialized in space instrumentation and remote sensing and GIS technology. He is an author of 16 books published by the European and United States famous publishers and more than 130 scientific papers.