JOURNAL OF TECHNIQUES

Journal homepage: http://journal.mtu.edu.iq

REVIEW ARTICLE - MANAGEMENT

Single Objective Optimization Methods in Electrical Power Systems: A Review

Ali Abdulmunim Ibrahim Al-kharaz [1*], Ahmed Bahaaulddin A.Wahhab [1], Mohammed Fadhil Ibrahim [1], Shahab Abdulla [2]

[1] Technical College of Management - Baghdad, Middle Technical University, Baghdad, Iraq

[2] University of Southern Queensland, Australia

[*] Corresponding author E-mail: Ali.Al-kharaz@mtu.edu.iq

Article Info.

Article history:

Received
12 August 2022

Accepted
13 September 2022

Publishing
31 March 2023

Abstract

Although the scheduling of maintenance tasks for generators is not a new issue, it has recently attracted new attention due to the significant rise in demand for expanding power system size in modern power systems. Generator Maintenance Scheduling (GMS) is a nonlinear optimization problem, highly dimensional and constrained, and determines when power-producing units must undertake well-planned preventative maintenance. The objective function includes binary variables to indicate whether a generator is undergoing maintenance at a given time and is subject to several restrictions described in this paper. However, the biggest concern of GMS is to produce a precise timetable for preventive maintenance of generating units with low cost and high reliability. Despite that, regrettably, a large volume of research works has accomplished solutions towards a model of GMS with the consideration of either maximizing system reliability or minimizing operation costs as an objective of their research work. This is called Single-Objective Problem (SOP), which involves one objective function that needs to be optimized. SOP is solved by Single-Objective Optimization Method (SOOM). The primary purpose of the research is to present a review of SOOM methods used in solving GMS problems.

Publisher : Middle Technical University

Keywords: Single-Objective Optimization; Generator Maintenance Scheduling; Heuristic; Fuzzy Logic; Genetic Algorithms.

1. Introduction

The availability of efficient, secure, and affordable electrical power is crucial to the sustainability of modern communities. Energy facilities have grown to be one of the most significant resources in a country's economy, necessitating effective operational planning. This planning of operations is regarded as a highly challenging undertaking, especially for emerging countries, because of the increased need for energy supply in those nations due to rapid development and economic demand [1]. This puts additional strain on developing nations' ability to balance their budgets because cleaner energy sources are more expensive than conventional methods of producing electricity [2]. Maintenance of a generator is one of the essential elements of operations and planning in power production systems, scheduling greatly impacts credibility, economic aspects, and reliable operation of a power system [3]. Software-based methods for resolving operational problems have been developed as a result of the complexity of electric power systems. One of the problems is the generator maintenance schedule (GMS) [4]. GMS's primary goal is to give power-generating units an accurate timeline for the effectiveness of preventative maintenance. That significantly contributes to increasing the system's reliability, minimizing the cost of operations, and extending the lifetime of power-generating units [5]. A good maintenance schedule plays a significant role in an electrical power system's economical and sustainable operation. Fig. 1 demonstrated the relationship between the GMS depends on maintenance duration in electrical power systems.

The biggest goal of GMS is to create a precise schedule for the routine maintenance of highly reliable and inexpensive generating units [6]. However, a large volume of research works has accomplished solutions towards GMS with the consideration of models either maximizing system reliability or minimizing operating costs as an objective of their research work [7]. The selection of particular objectives (for example, maximization of reliability and minimization of cost) and constraints (for example, load demand constraints and exclusion constraints) relies on several constructs. The reliability requirement has been defined in a variety of ways, including the sum of squared reserves which is the most common reliability criterion and loss of load probability predicted energy not supplied. The objective function is formulated in equation (1)[8]:

$$\min_{x_{i,t}} \left\{ \sum_{t=1}^{T} \left(\sum_{i=1}^{I} Gmax_{i,t} - \sum_{i=1}^{I} \sum_{t=1}^{T} Gmax_{i,t} \times x_{i,t} - D_t \right)^2 \right\} \quad (1)$$

Where $\sum_{i=1}^{I} Gmax_{i,t}$ represents the total generation capacity of the power system for period t, and component $\sum_{i=1}^{I} \sum_{t=1}^{T} Gmax_{i,t} \times x_{i,t}$ represents the generation capacity in scheduled maintenance in period t, and D_t is anticipated load demand for period tHence, component $\left(\sum_{i=1}^{I} \mathrm{Gmax}_{i,t} - \sum_{i=1}^{I} \sum_{t=1}^{T} \mathrm{Gmax}_{i,t} \times x_{i,t} - D_t\right)$ gives the reserve level in period t.

Nomenclature & Symbols

SOP	Single-Objective Problem	MC	Multi-Criteria
SOOM	Single-Objective Optimization Method	MO	Multi-Objective
MILP	Mixed-integer Linear Programming	MS	maintenance scheduling
DP	Dynamic Programming	MIP	Mixed-integer Programming
B&B	Branch and Bound	DE	Differential Evolution
IP	Integer Programming	SI	Swarm Intelligence
ACO	Ant Colony Optimization	HMs	Hybrid metaheuristics
TS	Tabu Searches	HDE	Hybrid Differential Evolution
PSO	Particle Swarm Optimization	GAs	Genetic Algorithms
SA	Simulated Annealing	ML	Machine learning
DPSA	Dynamic Programming with Successive Approximations EAs Evolutionary Algorithms		

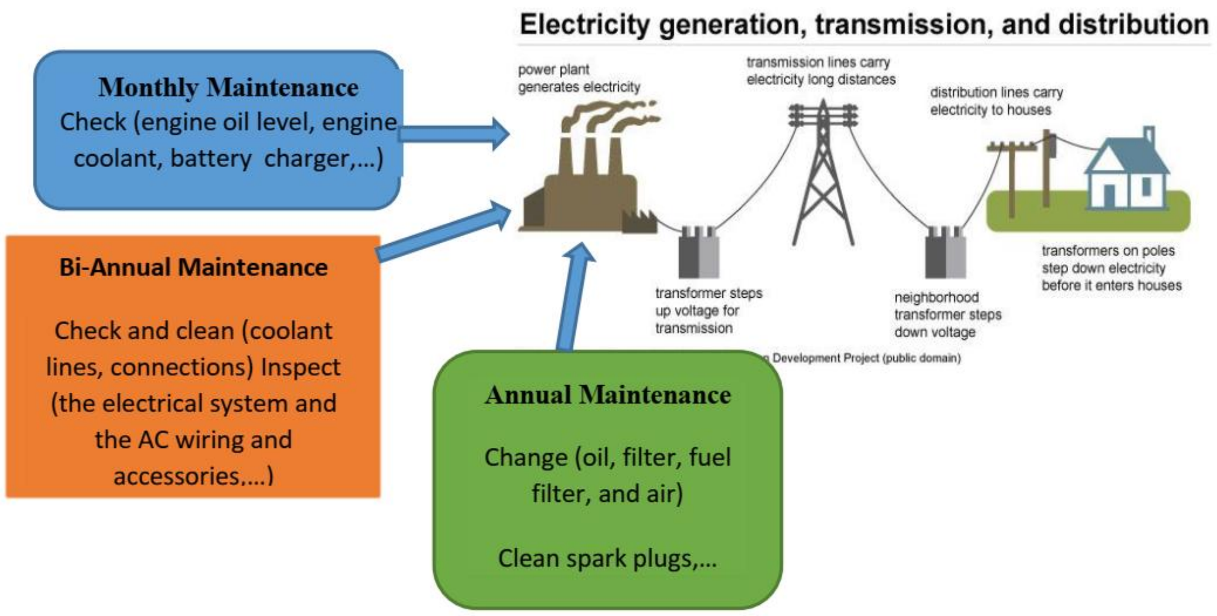
Electricity generation, transmission, and distribution
Monthly Maintenance
Check (engine oil level, engine coolant, battery charger,...)
Bi-Annual Maintenance
Check and clean (coolant lines, connections) Inspect (the electrical system and the AC wiring and accessories,...)
Annual Maintenance
Change (oil, filter, fuel filter, and air)
Clean spark plugs,...
power plant generates electricity
transmission lines carry electricity long distances
distribution lines carry electricity to houses
transformer steps up voltage for transmission
neighborhood transformer steps down voltage
transformers on poles step down electricity before it enters houses
Development Project (public domain)

Fig. 1. Relationship between electrical power systems and maintenance according to their duration

Minimizing the costs related to running a power plant is the economic cost objective. These costs include the cost of maintenance, startup, and electricity generation in the model that was developed. Because the fuel cost is the most dominant in the operation of a generator. Some literature has taken the fuel cost only in the economic cost objective that is given in equation (2)[9]:

$$F_t = \sum_{i=0}^{n} F_i(P_i)$$
$$F_i\,(P_i) = a_i + b_i\,P_i + c_i P_i^2 \qquad (2)$$

where F_t is the system fuel cost, F_i is the fuel cost of the *i*th unit, and a_i, b_i, and c_i are the coefficients related to the *i*th unit fuel. Parameter P_i represents the ith plant's generated power and n is the number of the last power unit of the system.

Such constructs are the GMS, data availability, solution accuracy, and the methodology to accomplish the intended solution [2,3, 10]. However, to overcome the problem of single-objective GMS models, Single-Objective Optimization Methods (SOOMs) are applied. SOOMs like exact algorithm, heuristic, metaheuristic, and hybrid metaheuristic; each one of these methods has its advantage and disadvantages. The exact algorithms or mathematical programming techniques such as Mixed-integer Linear Programming (MILP), Dynamic Programming (DP), branch and bound, Integer Programming (IP), and decomposition approach; obtains solutions by mathematical programming techniques are guaranteed to produce optimal solutions [2, 10]. However, these techniques require more comprehensive implementation, especially for realistic-sized instances of GMS problems [10]. Another technique is heuristics which is considered relatively enough for easy implementation and somewhat requires a short computation time. However, these techniques failed to find a solution near the optimal for the GMS problem [5]. Therefore, modern metaheuristics techniques were later introduced, and they surpassed heuristics in their ability to produce better (although not necessarily optimal) solutions [10].

Examples of metaheuristics to solve the GMS problem include Ant Colony Optimization (ACO), Tabu Searches (TS), Particle Swarm Optimization (PSO), Genetic Algorithms (GAs), and Simulated Annealing (SA) [2,10]. In general, these techniques use an intelligent way and make an effort to inspect various aspects of the search domain related to the optimization problem. Moreover, they seek to detect a close-optimal solution at a lower computation time and memory resources [10].

Despite the popularity of these techniques, there are still, in some cases, techniques that need to be combined to generate robust and effective optimized solutions. Here, the concept of hybrid metaheuristics has appeared. Compared to stand-alone methods, the hybrid approach has significantly outperformed them in various fields, solving many challenging optimization problems [11]. For example, hybrid SA and ACO algorithms and hybrid PSO by adding GA mutation operators are used to solve instances of GMS problems [12]. Furthermore, hybridization is not limited to combining different metaheuristic algorithms. Still, it can be enhanced by combining other optimization methods with metaheuristics, such as hybrid Differential Evolution, by incorporating the lambda iteration approach into the Differential Evolution procedure [13].

In the literature, a single criterion is inadequate for optimizing real GMS problems, especially for systems with several hard constraints. Hence, a Multi-Criteria (MC) or Multi-Objective (MO) solution is required to handle the GMS problem. For these reasons, the main purpose of this review is to analyze the previous work done by researchers related to single-objective optimization methods in solving the GMS problem. Accordingly, this paper is organized into sections. These concepts inform how this review is organized. The approaches to the GMS problem listed in Section 2 (numbers 2 to 6) are summarized together with some essential benefits and drawbacks of each technique. The associated works on these strategies from 2000 to 2019 are described in Section 3 (number 7), and the most pertinent conclusions are drawn in Section 4 (number 8).

2. Single-Objective Optimization Methods

Optimization is used to find the optimal values for the variables in a specific problem to maximize or minimize an objective function. In a fair length of time, appropriate GMS model solution methods are projected to produce either satisfactory or optimal solutions to practically significant model instances [2]. The dominating scheduling criterion in the objective function is primarily included in GMS models, which are often characterized as single optimization problems; nevertheless, some authors have included additional standards as restrictions. Extensive studies have been conducted to solve multi-objective energy-related problems, but only a few involve multi-objective GMS problems [10]. Consequently, there are several categories containing several methods. Each method has been successful in solving GMS problem. Thus, in this section attempt has been made to provide a brief review of the solution methods documented in the previous studies for solving single-objective GMS problems.

3. Mathematical Programming Techniques

Single-objective cases connected to the GMS problem are typically solved using mathematical programming approaches or exact algorithms, as referred to in specific literature [10]. The GMS problem's nature might make it simpler to formulate as a mathematical issue [2] that can be solved using an exact technique, such as Branch and Bound (B&B), Mixed Integer Linear Programming (MILP), Integer Programming (IP), the decomposition method, or Dynamic Programming (DP).

There has been a preference for solutions produced using mathematical programming approaches over those acquired using other techniques (such as metaheuristics), as the former provides a higher level of assurance that the solutions would be optimal [2, 10]. However, an issue with mathematical programming methodologies has been found in the time it takes for implementation processes, particularly in actual-sized GMS problems [7, 10]. It was also discovered that mathematical programming techniques were challenging to use in situations involving multiple scheduling criteria (such as for multiple conflicting objectives) [10].

3.1. Integer programming and mixed integer linear programming methods

Considered integer programming (IP) is one of the correct optimal methods applied for maintenance scheduling (MS) using an integer programming solver and a mathematical programming language [14]. The IP method is simple and can be implemented immediately. However, it has a drawback as it uses approximations in accounting for uncertainties and, in most instances, exceeds the capabilities of computer applications. To minimize the burdens of Integer Programming (IP) for computers, Mixed Integer Linear Programming (MILP) has been applied [14]. MILP is computationally efficient with a simple approach. However, MILP does not solve the problem of IP. Both integer programming and mixed integer linear programming is only applied in solving considerably small problem instances to optimality within an acceptable period, and this happens because the computational burdens are high [2]. In an attempt to eliminate the deficiencies associated with IP and MILP, the application of decomposition as a good branch and bound approach is mostly employed [2,10, 14].

3.2. Branch-and-bound method

Land and Doig were the first to introduce the Branch-and-Bound approach in 1960 [2]. It is a general algorithm method that can be applied to solving IP and MILP problems [2, 10]. The B&B method can provide an optimal solution and be found to be an acceptable computational process [14]. The method can identify an optimal solution for an instance of a combinatorial optimization problem through a systematic list of potential solutions among feasibly suitably selected sub-problems [2]. The succession of the search for an optimal resolution is represented by each set of probable solutions producing a rooted sub-tree that resembles a tree in a data structure (referred to as a search tree). These sub-trees structures are considered distinct but exclusive solution domain subsets. Comparisons of solutions uncovered in different branches of upper and lower boundaries search global tree estimates of values for optimal solution function is made through the algorithm. Finally, a judgment is made regarding whether to keep or throw away branches depending on whether they offer improved solutions in comparison to the finest solution that is contained too far inside the tree [2]. However, extra efforts have also been undertaken in situations with numerous objectives to solve GMS issue instances simultaneously utilizing the B&B technique.

3.3. Benders decomposition method

Given the exponential growth of their computation times relative to the size of the problem, it is not desirable to apply Mixed Integer Programming (MIP) solution techniques directly [15]. Decomposition methods may often be employed to address this concern [10]. Benders decomposition method is the most popular among various methods of decomposition techniques. Benders postulated this decomposition technique in 1962 [10, 15]. It allows a significant problem to be partitioned (decomposed) into the least number of problems of the same structure. Following the decomposition of the problems makes solving more minor problems much easier as they require a shorter time [2, 10]. This attribute makes the Benders decomposition technique a viable approach that provides a more straightforward solution when looking for a solution to address combinatorial optimization problems of a larger size. Eventually, Benders decomposition has become a viable solution approach for instances of significant GMS problems [2]. Considering intrinsic two-stage structure in MS problems in power systems, Benders decomposition has been considered a potential solution [15]. In this situation, the main problem is concerned only with those constraints associated with the MS problem and the resources required. In contrast, those of fuel management, as well as load and network constraints, may be shifted to the sub-problems [2,10]. However, in general, the disadvantage of the Benders getting the solution is not necessarily feasible and optimal [16].

3.4. Dynamic programming and dynamic programming with successive approximations methods

Dynamic Programming represents a process in which a big impractical, temporal decision problem is divided into an adequate number of least problems that overlap into sub-problems, thus, becoming very easy to solve. The partitioned smaller sub-problems obtained through DP are solved optimally and then utilized for solving more significant problems [2]. Breaking down the large problem into several sub-problems is expected to be made sufficiently so that such smaller sub-problems can be resolved separately within a more acceptable time frame than the time it may take to solve the large problem [2]. Rarely is the standard DP method directly used to provide a solution to a realistic GMS model instance [2]. Therefore, DP possesses a dimensional barrier that restricts its utilization for a small system [2, 14]. Eventually, the DP method was modified and called Dynamic Programming with Successive Approximations (DPSA), wherein an instance's dimensionality has been reduced [2, 14]. Adversely, the application of this adapted method is associated with disadvantages. However, it has the potential to converge to a solution in a faster way compared to the original DP approach (i.e., a reduced time of computation). However, providing a globally optimal solution cannot be guaranteed [2]. Consequently, direct usage of dynamic programming is not controllable, considering its computational requirements. This results in the application of DPSA to minimize the dimensions of the problem. Although the adapted method converges, however no guarantee for a global optimum.

4. Fuzzy Logic Approach

A mathematical model called fuzzy-set theory was put forth to examine the fuzziness of human cognitive processes. The theory is concerned with the extent of truth associated with the fact that a given outcome can be said to belong to a specific category. These techniques for solving the problem are limited to a setting of many-valued logic connected to approximations. Instead of providing binary (i.e., yes or no) answers, the approaches lead to conclusions that exhibit a level of conviction in the truth. A response is typically given this level of truth in consideration of a membership function with a zero to one range. The membership function of fuzzy numbers serves as the basis for fuzzy logic [2].

It has already been proven that GMS problems may have several inconsistent objectives. Moreover, there is an associated uncertainty in GMS resulting from the number of maintenance windows, workforce, and other resources, as well as uncertain load demand [2]. Thus, the introduction of fuzzy sets addressed both the objective function and concerned constraints, considering that a fuzzy environment can deal with a variety of objectives as well as uncertainties in a number of the constraints. Using the membership functions used to solve the model instance, the objectives and soft constraints of the GMS model instance are extracted. The optimal maintenance scheduling is then obtained by the interaction of the various fuzzy sets, representing the objective function and the constraints. Fuzzy mathematical programming techniques are applied in solving fuzzy GMS model instances or sometimes through metaheuristics [2]. Fuzzy logic can approximate functions and simulate any continuous function or system, and it has the advantages of being easy to understand, adaptive, and tolerant of poor data. Even nonlinear functions of any complexity can be modeled using it. It was found, however, that developing a fruitful fuzzy system is not always as easy as it first seems. The membership function and rules of fuzzy logic are created by trial and error. It takes a lot of effort to find the right membership function and rules for complex systems [17]. The optimal maintenance scheduling is then obtained by the interaction of the various fuzzy sets, representing objective function and the constraints.

5. Heuristic Search Algorithms

Heuristic was derived from a Greek word called heuristikein, meaning to uncover or find [2]. A heuristic presents any given method of problem-solving, discovery, or learning that applies a practical approach that is not guaranteed to be optimal but adequate in achieving a proximate target [10]. Heuristic searches typically employ a continuous trial-and-error solution technique for an acceptable amount of time to identify solutions that aren't necessarily ideal but are at least sensible [2]. Heuristic techniques are problem-dependent and are designed and usable for a specific situation. It was also reported that heuristic solution approaches are also applicable in solving instances of single-objective GMS models (Charest and Ferland) [18]. Heuristic solution approaches were relatively simple to implement and possess a short computational time frame; however, the solutions they yield are primarily of relatively poor quality [2]. And in most instances, they are seldom applied in solving GMS problems [10].

6. Metaheuristic Search Algorithms

Metaheuristics are thought to be superior to heuristics in several ways, including flexibility and the potential to produce solutions of higher quality, as suggested by their name [2]. Metaheuristics are classified into two broad categories; population-based metaheuristics and trajectory-based metaheuristics [2, 10]. The trajectory-based metaheuristics category is considered a single solution that can be manipulated during the search. Examples of this may include algorithms that cover local search algorithms, TS, and the approach of SA. In the latter category, population-based metaheuristics, a manipulation is made for multiple candidate solutions through a simultaneous process [2]. Examples include evolutionary algorithms, swarm optimization techniques, and scatter searches [2].

Metaheuristics are primarily applied when the dimensions of a GMS problem increase so that exact solution methodologies may require a highly extended period to implement. Literature documents that metaheuristics techniques mainly gain good (not essentially optimal) solutions within a reasonable computational time frame [10]. Recently, it was discovered that the use of metaheuristic approaches to solve quick computation of GMS problems that were extremely close to optimal [10]. In conclusion, metaheuristics can be used to tackle various optimization issues. They are flexible enough to address any optimization problem. By examining the frequently vast solution search space of problems thought to be problematic in general, metaheuristics can solve specific examples of those problems. These algorithms achieve this by effectively examining the space while compressing its adequate size. Metaheuristics are mainly used to gain resilient algorithms, solve large issues quickly, and solve little problems quickly.

Additionally, they are very flexible and straightforward to design and use. Several popular metaheuristic techniques used in GMS problem-solving include Ant Colony Optimization (ACO), Tabu Searches (TSs), Genetic Algorithms (GAs), Simulated Annealing (SA), and Particle Swarm Optimization (PSO) [2, 10].

6.1. Evolutionary algorithms

Evolutionary Algorithms (EAs) are a significant subcategory of metaheuristic algorithms [19]. Comparable to natural processes like crossover, mutation, and selection found in live organics, EAs go through similar processes [20]. Several algorithms have been developed under the EA category, like genetic algorithms, genetic programming, evolutionary programming, and evolution strategies [20]. For example, the GMS problem has been solved using differential evolution and genetic algorithms in the literature [7, 13, 21, 22]. Some EAs, including genetic and differential evolution algorithms, are explained.

6.1.1. Genetic algorithms

John Holland developed Genetic Algorithm (GA) in 1975 [20]. A genetic algorithm is used as a population-based metaheuristic to imitate biological evolution and other natural selection processes to find appropriate answers [2,10]**.** Despite GA being easy to modify and reliable enough not to become trapped in local minima [23], finding a fitness function is not always easy**.** Genetic algorithms often converge prematurely to a solution [23]. There are three main operators to the GA: crossover, mutation, and selection [20,24].

The population, created randomly or using a heuristic algorithm, must first be initialized. The evaluation process is the second step in GA. In the context of a genetic algorithm, evaluation is an operator for calculating the quality of solution or fitness. For selection and replacement operators, solution fitness is necessary. In other words, the selection process favours solutions with higher fitness values. The loop statement with a terminating condition is used in the third phase. One or more conditions, such as the precise number of iterations, the precise execution time, and the largest number of iterations without any solution quality improvement, could be used to stop the execution. In the fourth selection procedure phase, a part of the population is chosen for mutation operators and crossover. However, there are several selection techniques, such as Tournament Selection, Linear-Rank Selection, and Roulette-Wheel Selection. The recombination or crossover operators are the fifth step. Crossover operators combine the genes from the selected solutions to produce a new solution. Crossover operator is of several types within the literature, some of which include one-point, two-point, uniform crossover, and N-point. Finally, the GA algorithm has a mutation operator in its sixth stage.

A mutation is called the process of perturbing the solution with a low probability. The algorithm will benefit from the mutation operator's assistance in avoiding early convergence. For example, the mutation process in a binary solution representation could be done by switching the value from 1 to 0 or vice versa. The bit-flip mutation is the name of this technique. The assessment of the new solutions is the seventh step. Similar to step two, this step replaces the entire population with new solutions. The replace operator is the final step of the GA algorithm, which replaces existing solutions from the population with newly generated ones. The application of the genetic algorithm is made to many forms of scheduling problems, which include but are not limited to transport systems and scheduling of production [25], manufacturing scheduling [26] as well as scheduling workflow applications applicable to cloud environments [27].

6.1.2. Differential evolution algorithm

Another type of optimization technique, Differential Evolution (DE), was first introduced in 1995 by (Storn and Price) [5, 7, 20]. It is a distinct population-based algorithm that uses the operators' crossover, mutation, and selection to evolve a population into potential contenders for an optimal solution [20]. Due to its simple evolutionary algorithm, characteristics can create offspring via the combination of a parent individual with many others of the same population [22,28]. Depending on its fitness, an offspring can replace the parent solution. However, the critical steps within DE remained crossover, mutation, and selection, as mentioned earlier. Being a dominant population-based evolutionary algorithm, DE can demonstrate exceptional performance in various problems in many fields of applications [5]. Thus, its application has been promising in solving several complex optimization problems. Thus, it has been confirmed as a successful algorithm that solves real-world optimization problems in different fields of applications [22]. Premature convergence has been reported to be the main drawback of conventional DE. However, premature convergence can be overcome by using a large population, which leads to a substantial computational effort. This approach allows DE to find a global optimal or approximately close to the global optimal solution for a specific problem under consideration. GMS problems can have many decision variables. Thus, obtaining optimum global values from those serious dimension problems, DE requires the deployment of a large population [5].

6.2. Local search algorithms

The local search algorithms were introduced on the excellent idea that solution modification can be made successively through performing moves capable of changing the solutions locally [20]. Neighborhood solutions are actions that can be taken to arrive at these solutions. The neighborhood of the answer has been searched using a variety of techniques, including random, iterated, greedy, variable neighbourhood search, and steepest descent algorithms [29,30]. However, metaheuristics algorithms for local search, including the simulated annealing method and tabu search, have been applied in solving GMS problem instances.

6.2.1. Simulated annealing

The simulated annealing (SA) method was evolved in 1983. It is a trajectory-based metaheuristic used to solve instances of optimization problems that depend on the metallurgy of the physical process of annealing [2]. SA was inspired by the metallurgical annealing process, in which a physical system finally reduced the energy state by heating it and allowing it to cool gradually [10]. The algorithm creates almost the answer by disturbing the existing one, just like the physical process. The close solution is regarded as the new current solution when it is more crucial than the existing answer in a certain situation (regarded as a hill-climbing algorithm). However, in other instances in which a nearby solution is found to be worse compared to the current solution, then, with a certain probability, the current is considered as the new current solution. The probability here depends on the likelihood of a worse nearby solution not being accepted, where the highly worse solution has a high probability of not being accepted than a slightly worse solution [10]. The temperature of the system is another parameter on which this probability depends. The existence of higher temperatures indicates a high likelihood of accepting non-improving solutions. Probabilistically, a poorer solution must be accepted for the algorithm to explore the search space and avoid becoming stuck in local optima. The temperature is gradually decreased over time with the expectation that the SA algorithm will eventually freeze upon finding a nearly optimal solution. Considering that no dynamic extraction of information is used during the search, the SA algorithm is categorized as a memoryless method [10]. SA algorithm has the advantage of being associated with achieving high-quality solutions. This attribute has made it universally applicable and considered easy to implement (encode), as well as flexible and allowing s adding constraints in a relatively straightforward manner [10]. However, SA still has certain drawbacks because it requires the user to specify a substantial number of algorithm parameter values. The parameters include the starting temperature, the rate of temperature decline, the duration of temperature stages, and the standards for search closure. Even though the extensive literature contains proposed values for the SA algorithm parameters, they are primarily problem-specific. Therefore, a researcher must always perform thorough empirical research to determine the ideal combination of these parameter values. The fact that the algorithm needs numerous iterations to provide a good result is another drawback of SA. The implication is that it takes a long time to compute, especially for cases where evaluating the values of objective function solutions takes a fair amount of time [10].

6.2.2. Tabu search

Glover was the first to introduce the Tabu Search (TS) algorithm in 1986, which is a form trajectory-based search metaheuristic. It borrows ideas from and incorporates components of human memory. To avoid having to look up previous answers, TS can keep track of the search's recent history using a short-term memory unit called a tabu list [10]. Through TS, it is possible to add a short-term memory of recently visited locations inside the search space. This list of short-term memories is known as the tabu list. For some iterations of the TS algorithm, regions or solutions in the tabu list might not be visited again after the previous visit [2]. Some of its approaches are similar to the SA algorithm, which means that it does not use memory to make learning.

Similar to SA methods, TS considers all potential alternatives to an existing solution. The existing solution is locally perturbed to achieve this. The best adjacent option is then selected in TS as the new current solution (a worsening solution may be accepted in cases where no improvement was identified in the vicinity of the current solution). However, after a predetermined amount of iterations, the search is not allowed to return to any of the tabu list's solutions (considered as the tabu tenure). Making sure that no algorithm gets stuck at a specific local optimal solution is crucial. The iteration of this method continues until a predetermined stopping criterion is met. The conditions of the algorithm can be altered before a solution is eliminated from the tabu list. This is because maintaining a solution on the list, especially for a limited amount of iterations, aids in narrowing the search space to the area where a reasonable solution has been found. The search can be expanded to other locations where the search space is studied, and prospective reasonable solutions are located when a solution is retained in the tabu list for several rounds [2]. The short searching time of TS is an advantage, but the limited global search capability is a drawback [31].

6.3. Swarm intelligence algorithms

Swarms are seen as "a set of individuals possessing independent individual dynamics but exhibiting intimately connected behaviors and collectively accomplishing some task," according to Gazi and Passino [32]. Artificial bee colonies, ant colony optimization, cuckoo searches, firefly algorithms, and particle swarm optimization are some examples of nature-inspired Swarm Intelligence (SI) algorithms [20, 24]. The biological behavior of some animals, such as bees, colonies of ants, bird flocks, and fish schools, is mimicked by some SI algorithms [20, 32]. The field of computing study, particularly optimization, is inspired by swarm intelligence algorithms [20, 33]. The swarm intelligence algorithm has certain principles, including adaptability, diverse response, proximity, quality, and stability [34].

Lim [34] documents the basic principles of swarm intelligence; (i) proximity, which is considered as the extent to which simple computation of time and space can be performed in response to environmental stimuli; (ii) quality: which represents the effort towards responding to quality factors relating to food and safety; (iii) diverse response: which represent to the extent to which the resources have been distributed for safeguarding changes against the environment; (iv) stability: which represent the extent to which group behaviour is maintained against every environmental fluctuation; and lastly (v) adaptability: which represents the extent to which the group behaviour is changed towards better environmental adaptation.

The literature documents the successful performance of Swarm intelligence methods in the context of scheduling, which is of outstanding achievement for science and the industry [20]. Exclusively, Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) have been the most successful swarm intelligence algorithms reported in the literature [10]. The following subsections explain the most popular methods of swarm intelligence algorithms.

6.3.1. Ant colony optimization

Ant Colony Optimization (ACO) is one of the population-based metaheuristics [2, 10]. ACO is related to the way ants naturally behave and interact with one another as they search for food. If a good food source is not found, ants frequently seek food initially using a random method without systematic coordination among themselves. A pheromone is released along an ant's path straight from the food source as it turns to the colony after locating a food supply. As the distance between the food source and the ant colony grows, the level of pheromone along the path steadily drops. After additional ants detect it, the pheromone level eventually drops as expected.

On the other hand, if there is a proximity between the food supply and the ant colony, the pheromone level rises quickly after additional ants become aware of it. Ants eventually begin to go along that route in search of food. Ants emit pheromones as they go along the trail, and these releases raise the concentration of pheromones there. Since ants randomly chose a path in a biased order toward paths with high levels of pheromone, such results in successfully identifying shorter paths by ants towards their nest. Consequently, ACO was introduced to identify the shortest paths in graphs. However, it was applied along the line to solve other optimization problems in scheduling and assignment.

The ACO algorithm is an interaction of three basic procedures relating to constructing solutions, daemon actions, and updating pheromones. The construct solutions procedures refer to the routines required for ants to construct solutions incrementally. This means that following the starting point selection, a node is added at a time to the path of ants. The decision of where the ant will go next is biased based on the pheromone trails τ_{ij} and the information obtained by the ants on a possible food source η_{ij}. Generally, the higher the two values, the more the probability of deciding on the associated edge. Typically, two parameters $\alpha > 0$ and $\beta \geq 0$, are applied to gauge the associated power of the pheromone and the heuristic values, respectively. The ACO variant is defined by a specific rule associated with the ant's choice. The second procedure is daemon actions. It is a procedure that combines all problem-specific operations required to boost the ACO algorithm's performance. The best example of this procedure is the introduction of a local search phase. Additionally, this approach executes centralized tasks which an individual cannot perform. Though this procedure is optional, most daemon actions are essential to improve performance significantly [35]. The last procedure is updating pheromones. This approach uses two values to enable the update of the pheromone trail. The first is decreasing pheromone values through pheromone evaporation, which depends on a parameter $\rho \in [0,1]$, called the rate of evaporation. The pheromone evaporation is aimed at avoiding an unrestricted increase in pheromone values. It also allows the ant colony to forget poor earlier choices. The second is increasing pheromone values through pheromone deposits which belong to reasonable solutions generated by the ants. The amount of the solutions and pheromone deposited are peculiar to each ACO variant [35]. Due to its stochastic feature and iterative adaptation process based on positive feedback, which both enables the exploration of a sizable area of the search space, ACO has grown in significance [36]. Nevertheless, the basic ACO suffers from problems like stagnation behaviour and premature convergence [37], or ACO may suffer from slow convergence as in the [38]. Through searching for an optimal solution in a graph, In his Ph.D. dissertation, Marco Dorigo presented the initial ACO algorithm (Alobaedy) [20]. The variants of ACO are i) Ant System, ii) Elitist Ant System, iii) Rank-Based Ant System, iv) Max-Min Ant System, and v) Ant Colony System.

6.3.2. Particle swarm optimization

Another swarm intelligence algorithm is Particle Swarm Optimization (PSO), which was proved successful. It is also a population-based metaheuristic [10]. and Kennedy and Eberhar were the ones that initially developed this algorithm. It simulates the social behaviour of natural organisms (particularly the movements). More specifically, that of flocks of birds or groups of fish while looking for food sources through information exchange.

The PSO algorithm is often started with a random population of solutions. The population is updated each time the algorithm iterates based on potential solutions to some problem optimization instances discovered by other members of such a population through the search for optimal solutions. A particle is any solution that an individual finds themselves with it. Particles are "moved" using solution space, but this movement is biased toward the best particles already present in the system. The objective function determines the related fitness and the associated coordinates for each particle in a solution space. In addition, the particles maintain track of those who were a part of the best solution discovered by the entire swarm or flock of particles before the search, with the majority of the search direction headed in that direction. The attribution of individual velocity is each particle, updated at the time of each iteration to produce acceleration to the direction of reasonable solutions. Usually, the velocity of a particle is weighted its changes by its current location and relative to the best solution's current location in the solution space [2]. This method was initially proposed exclusively for continuous optimization problems; however, it was later modified for application in binary and integer optimization problems [10].

Despite that, in its standard form, PSO performs simply in instances of convex optimization problems involving restricted search space. However, it was found to fail in more complicated non-convex optimization problems associated with multi-minima functions. Eventually, it was less efficient in exploring reliable search space that ensures high-quality solutions. Hence, it is not easy to guarantee the identification of the global optimum solution through PSO [39].

6.4. Hybrid metaheuristics

The term "hybrid" describes combining algorithms to integrate each other to improve performance. Hybrid algorithms are combinations of two or more algorithms that function well together and complement one another. The term "hybrid metaheuristics" is widely used to describe these algorithms (HMs) [40]. Numerous algorithms can be hybridized, including heuristic and metaheuristic [20,41]. As a result, some techniques employ a hybrid metaheuristic for a maintenance schedule, including a hybrid Simulated Annealing (SA) and Ant Colony Optimization (ACO) algorithm and a hybrid Particle Swarm Optimization (PSO) by including a mutation operator of GA. [12]. Furthermore, hybridization is not limited to combining different metaheuristic algorithms. It combines other optimization methods, such as Hybrid Differential Evolution (HDE), by incorporating the lambda iteration approach into the DE procedure [13].

Existing empirical evidence revealed the hybrid approach was efficient and more reliable compared to other stand-alone metaheuristic approaches [12, 42]. In general, hybridization can improve either the accuracy or the computation speed. [43]. However, there are disadvantages and challenges in using hybrid algorithms. In terms of algorithm architecture, the hybridization process typically results in adding new components to the hybrid algorithm's overall architecture. This makes the hybrid algorithm more complex. Hybrid algorithms also have other issues. For instance, most hybrid algorithms will add more parameters, making it more challenging to fine-tune their parameters. Additionally, a hybrid's complex structure typically makes it more challenging to analyze, providing little insight into the mechanisms underlying its effectiveness. Moreover, hybrid algorithms are a little more challenging to implement and, therefore, more prone to mistakes [43].

7. Results and Analysis

7.1. Related works

This section presents research that provides solutions to the GMS problem depending on a single objective function from the year 2000 to the year 2019. Fig. 2 shows the trend of studies related to Mathematical Programming Techniques used for solving GMS. [44] proposed Mixed Integer Programming (MIP), and [45] proposed Mixed Integer Linear Programming (MILP). [46] proposed Dynamic Programming (DP), [47] proposed Dynamic Programming and Successive Approximations (DPSA) method. [48] proposed Branch and Bound (B&B) method. And finally, [49, 50] proposed the Benders decomposition method. Previous studies show that using the Benders decomposition method was higher than other mathematical programming methods because of its ability to solve significant problems within an acceptable, reasonable time. Despite mathematical programming techniques giving a guarantee in producing optimal solutions but one problem with mathematical programming methods is how long the implementation procedures take, especially when dealing with GMS problems of realistic size [51].

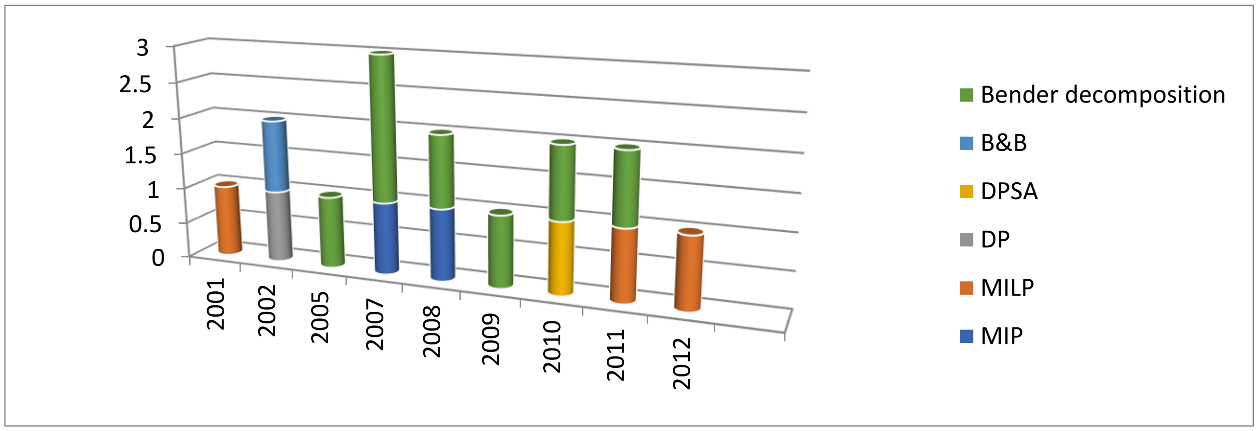

Fig. 2. Trend of using mathematical programming techniques

Fig. 3 shows the trend of studies related to heuristic, metaheuristic, and hybrid metaheuristic search algorithms used to solve single-objective GMS instances.

[48] proposed depth-first branch and bound search strategy for heuristic search algorithms. [52] combined greedy and constructive algorithms. [53] Presented a problem-solving method that combines constraint programming formulation with many heuristics.

For metaheuristic search algorithms: in evolutionary algorithms, [21] proposed genetic algorithm optimization techniques. [54] proposed differential evolution algorithms. In local search algorithms, [55] proposed Parallel-Refined Simulated Annealing (PRSA). And [56] proposed Simulated Annealing (SA). Whilst [57] proposed Tabu Search (TS) algorithm. And in Swarm Intelligence Algorithms, [58,59] proposed the ACO algorithm and its variants. When [60] proposed a Particle swarm optimization algorithm. Recently, ant colony optimization has been applied in solving of GMS problem that is too close to optimality in an acceptable computational time [61].

For hybrid metaheuristic search algorithms: [62] proposed hybrid GA/simulated annealing (SA) techniques. By employing the GA's adding mutation operator to modify the conventional particle swarm optimization algorithm. [63] developed a particle swarm optimization technique with a hybrid differential evolution. Extremal optimization EO&GA serves as the local search in the [64] proposed hybrid evolutionary algorithm, which uses GA as the evolutionary algorithm. [65] proposed hybrid local search/SA algorithm. [66] The suggested hybrid particle swarm optimization-based genetic algorithm and optimization-based evolutionary programming are used to tackle the long-term generation maintenance scheduling problem. By incorporating the lambda iteration approach into the DE procedure, [7, 13] introduced hybrid differential evolution (HDE). [12] The proposed optimization approach is based on simulated annealing (SA) and ant colony optimization (ACO).

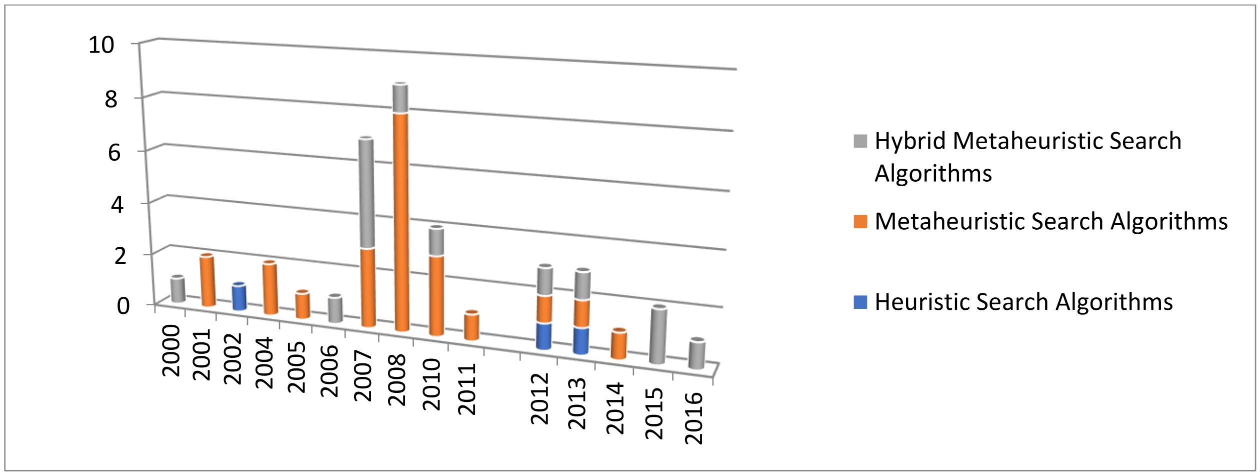

Fig. 3. Trend of using heuristic, metaheuristic, and hybrid metaheuristic search algorithms

For fuzzy system approach, as mentioned previously, can be applied in solving instances of GMS models with the aid of mathematical programming techniques or through metaheuristics. Fig. 4 shows the trend of studies related to fuzzy mathematical programming and fuzzy metaheuristic techniques used for solving Single-Objective GMS models. A fuzzy approach with metaheuristics has been applied to generator maintenance scheduling in the literature [67] which proposed fuzzy with evolutionary algorithms. [68] proposed fuzzy with the GA and SA algorithms hybrid. Whilst fuzzy approach with mathematical programming applied in the literature [69] which proposed fuzzy integer programming with Branch and Bound method. However, in the literature, fuzzy metaheuristic techniques perform better than fuzzy mathematical programming techniques because of the ability of metaheuristics to solve high-dimensional problems within a reasonable computation time.

In recent years, Artificial Intelligence and machine learning (ML) techniques have become applied in various scopes. Several studies used (ML) to solve single-objective GMS problems in electrical power systems, see Table 1. For example, reinforcement learning (RL) and teaching–learning-based optimization (TLBO) [70,71].

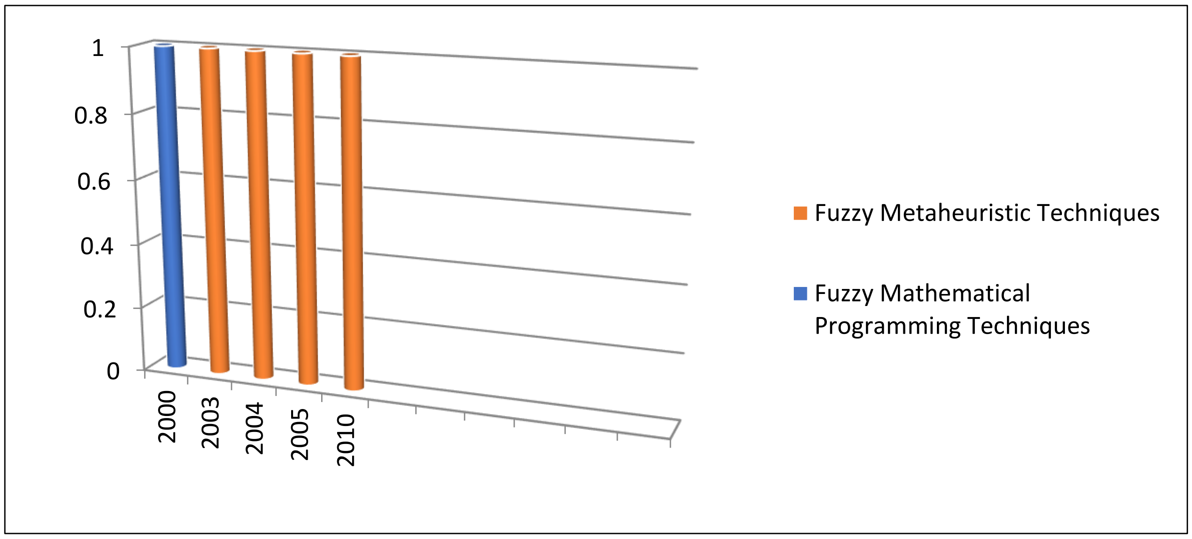

Fig. 4. Trend of Using Fuzzy Mathematical Programming Techniques& Fuzzy Metaheuristic Techniques

Table 1. Shows the summary of some previous studies for solving single-objective GMS problems

No.	Method	Advantages	Disadvantages
1	IP	Simple, can be implemented immediately.	It exceeds the capabilities of computer applications and uses approximations to account for uncertainties.
2	MILP	Efficient and simple.	Computational burdens are high.
3	Benders decomposition	Divide the big problem into minor problems that have much easier to solve as they require a shorter time.	Getting the solution is not necessarily feasible and optimal.
4	DL	Solving a practical large system with DP requires excessive computer time and storage, which may not be operationally acceptable.	The computational time required.
5	Fuzzy logic	It has been easy to understand, adaptive, and tolerant of poor data.	It takes a lot of effort to find the right membership function and rules for complex systems.
6	Heuristic methods	Clear and simple concept, and low computational time.	The solutions they yield are primarily of relatively poor quality. In most instances, they are seldom applied in solving GMS problems.
7	SA algorithm	It's associated with achieving high-quality solutions. Easy to implement (encode), as well as flexible, and allowing s adding constraints in a relatively straightforward manner.	Needs numerous iterations to provide a good result. The implication is that it takes a long time to compute.
8	Tabu search	It has a fast search speed and produces diversified solutions.	Limited global search capability.
9	Hybrid metaheuristics	Efficient and more reliable compared to other stand-alone metaheuristic approaches. In general, hybridization can improve either the accuracy or the computation speed.	Adding new components to the hybrid algorithm's overall architecture. This makes the hybrid algorithm more complex. most hybrid algorithms will add more parameters, making it more challenging to fine-tune their parameters.

8. Conclusion

This paper addressed single - objective GMS problem and the single-objective optimization methods used for solving this problem. However, a single objective is insufficient to specify the optimization of the actual GMS problem, so a multi-objective GMS model is required, which includes multiple objective functions that must be optimized simultaneously. In addition, some modifications in single-objective optimization methods are needed to be more relevant in solving instances of multi-objective GMS models.

Acknowledgement

In addition to acknowledging Middle Technical University, the authors would like to thank the Technical College of Management-Department Baghdad's of Information Technology, particularly assistant professor Ali Sami Aziz, for his cooperation and help in finishing our research.

References

[1] J. Eygelaar, D. P. Lötter, and J. H. van Vuuren, "Generator maintenance scheduling based on the risk of power generating unit failure," Electrical Power and Energy Systems, vol. 95, no. 2018, pp. 83–95, 2018.
[2] J. Eygelaar, "Generator maintenance scheduling based on the expected capability of satisfying energy demand," Doctoral dissertation, Stellenbosch: Stellebosch University, 2018.
[3] Y. Lee, J. Choi, J. Cha, and M. Jung, “Smarter Visual System of Generator Maintenance Scheduling Including Multi-Objective Functions by GA,” IFAC-PapersOnLine, vol. 49, no. 27, pp. 212–217, 2016, doi: 10.1016/j.ifacol.2016.10.685.
[4] El-Amin, I., 2000. Electric generator maintenance scheduling. Maintenance, Modeling and Optimization, pp.55-80.
[5] G. Balaji, R. Balamurugan, and L. Lakshminarasimman, "Fuzzy clustered multi-objective differential evolution for thermal generator maintenance scheduling," International Journal of Intelligent Engineering and Systems, vol. 9, no. 1, pp. 1–13, 2016, doi: 10.22266/ijies2016.0331.01.
[6] B. Lindne, R. Brits, J. van Vuuren, and J. Bekker, "Tradeoffs between levelling the reserve margins and minimizing production cost in generator maintenance scheduling for regulated power systems," International Journal of Electrical Power and Energy Systems, vol. 101, pp. 458–471, 2018, doi: 10.1016/j.ijepes.2018.02.018.
[7] G. Balaji, R. Balamurugan, and L. Lakshminarasimman, "Mathematical approach assisted differential evolution for generator maintenance scheduling," International Journal of Electrical Power and Energy Systems, vol. 82, pp. 508–518, 2016, doi: 10.1016/j.ijepes.2016.04.033.
[8] Moyo, L., Nwulu, N.I., Ekpenyong, U.E. and Bansal, R.C., 2021. A tri-objective model for generator maintenance scheduling. IEEE Access, 9, pp.136384-136394.
[9] Ghorbani, N. and Babaei, E., 2016. Exchange market algorithm for economic load dispatch. International Journal of Electrical Power & Energy Systems, 75, pp.19-27.
[10] B. G. Lindner, "Bi-objective Generator Maintenance Scheduling for a National Power Utility,",2017.
[11] C. Blum and G. R. Raidl, Artificial Intelligence : Foundations, Theory, and Algorithms Hybrid Metaheuristics Powerful Tools for Optimization, 2016.
[12] N. Bali and H. Labdelaoui, "Optimal Generator Maintenance Scheduling Using a Hybrid Metaheuristic Approach," International Journal of Computational Intelligence and Applications, vol. 14, no. 02, pp. 1550011–1 to 1550011–11, 2015, doi: 10.1142/S146902681550011X.
[13] G. Balaji, R. Balamurugan, and L. Lakshminarasimman, "Generator maintenance scheduling in power systems using metaheuristic-based hybrid approaches," ARPN Journal of Engineering and Applied Sciences, vol. 10, no. 22, pp. 10566–10577, 2015.
[14] A. Ahmad and D. P. Kothari, "A REVIEW OF RECENT ADVANCES IN GENERATOR MAINTENANCE SCHEDULING," Electric Machines & Power Systems, vol. 26, no. 4, pp. 373–387, 1998, doi 10.1080/07313569808955829.
[15] M. Gendreau, J. E. Mendoza, and L. Rousseau, "Maintenance Scheduling in the Electricity Industry : A Literature Review Aurélien Froger Maintenance Scheduling in the Electricity Industry : A Literature Review," no. October 2014.
[16] G. R. Raidl, T. Baumhauer, and B. Hu, "Boosting an exact logic-based benders decomposition approach by variable neighborhood search," Electronic Notes in Discrete Mathematics, vol. 47, pp. 149–156, 2015, doi: 10.1016/j.endm.2014.11.020.
[17] R. Singh, A. Kainthola, and T. N. Singh, "Estimation of elastic constant of rocks using an ANFIS approach," Applied Soft Computing Journal, vol. 12, no. 1, pp. 40–45, 2012, doi: 10.1016/j.asoc.2011.09.010.
[18] M. Charest and J. A. Ferland, "Preventive maintenance scheduling of power generating units," Annals of Operations Research, vol. 41, no. 1993, pp. 185–206, 1993, doi: 10.1007/BF02023074.
[19] X. Yu and M. Gen, Introduction to Evolutionary Algorithms. 2010.
[20] M. M. Alobaedy, "Hybrid Ant Colony System algorithm for static and dynamic job scheduling in grid computing," UUM, 2015.
[21] A. Volkanovski, B. Mavko, T. Boševski, A. Čauševski, and M. Čepin, "Genetic algorithm optimization of the maintenance scheduling of generating units in a power system," Reliability Engineering and System Safety, vol. 93, no. 6, pp. 779–789, 2008, doi: 10.1016/j.ress.2007.03.027.
[22] G. Balaji, R. Balamurugan, and L. Lakshminarasimman, "Reliability based Generator Maintenance Scheduling using Integer Coded Differential Evolution Algorithm," International Journal of Computer Applications, vol. 131, no. 6, pp. 27–38, 2015, doi: 10.5120/ijca2015907473.
[23] P. G. Majeed and S. Kumar, "Genetic algorithms in intrusion detection systems: A SURVEY," 2014.
[24] X. S. Yang, Nature-Inspired Optimization Algorithms, no. March. 2014.
[25] J. Hartmann, T. Makuschewitz, E. M. Frazzon, Bernd, and Scholz-Reiter, A genetic algorithm for the integrated scheduling of production and transport systems, vol. 45, no. 1. 2014.
[26] M. Gen, W. Zhang, L. Lin, and J. Jo, Recent Advances in Multiobjective Genetic Algorithms for Manufacturing Scheduling Problems. 2014.
[27] L. Singh and S. Singh, A genetic algorithm for scheduling workflow applications in unreliable cloud environment, vol. 335. 2014.
[28] R. Storn and K. Price, "Differential Evolution – A simple and efficient heuristic for global optimization over continuous spaces," Journal

ofGlobal Optimization, vol. 11, pp. 341–359, 1997, doi: 10.1071/AP09004.
[29] G. Zäpfel, R. Braune, and M. Bögl, Metahuristics Search Concepts. 2010.
[30] M. Gendreau and J.-Y. Potvin, Handbook of Metaheuristics, vol. 157. 2010.
[31] X. Zhang et al., "3D Protein structure prediction with genetic tabu search algorithm," BMC Systems Biology, vol. 4, no. SUPPL. 1, pp. 1–9, 2010, doi 10.1186/1752-0509-4-S1-S6.
[32] V. Gazi and K. M. Passino, swarm stability and optimization. 2011.
[33] C. Pintea, Advances in Bio-inspired Computing for Combinatorial Optimization Problems. 2014.
[34] C. P. Lim and L. C. Jain, innovations in swarm intelligence, vol. 91. 2009.
[35] L. Manuel, M. Maur, and M. M. De Oca, "Parameter Adaptation in Ant Colony Optimization Thomas St u," no. January 2010, doi 10.1007/978-3-642-21434-9.
[36] H. N. K. Al-Behadili, K. R. Ku-Mahamud, and R. Sagban, "Ant colony optimization algorithm for rule-based classification: Issues and potential solutions," Journal of Theoretical and Applied Information Technology, vol. 96, no. 21, pp. 7139–7150, 2018.
[37] R. Gan, Q. Guo, H. Chang, and Y. Yi, "Improved ant colony optimization algorithm for the traveling salesman problems," Journal of Systems Engineering and Electronics, vol. 21, no. 2, pp. 329–333, 2014, doi: 10.3969/j.issn.1004-4132.2010.02.025.
[38] A. M. Jabbar, K. R. Ku-Mahamud, and R. Sagban, "Ant-based sorting and ACO-based clustering approaches: A review," in In 2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), 2018, no. July, pp. 217–223, doi: 10.1109/ISCAIE.2018.8405473.
[39] G. Abbas, J. Gu, U. Farooq, A. Raza, M. U. Asad, and M. El-Hawary, "Solution of an economic dispatch problem through Particle Swarm Optimization: A detailed survey – part II," IEEE Access, vol. 5, pp. 24426–24445, 2017, doi: 10.1109/ACCESS.2017.2768522.
[40] X. Yang, Recent Advances in Swarm Intelligence and Evolutionary Computation. 2015.
[41] E.-G. Talbi, A Unified Taxonomy of Hybrid Metaheuristics with Mathematical Programming, Constraint Programming and Machine Learning, 2013.
[42] Y. Su and R. Chi, "Multi-objective particle swarm-differential evolution algorithm," Neural Computing and Applications, vol. 28, no. 2, pp. 407–418, 2015, doi: 10.1007/s00521-015-2073-y.
[43] T. O. Ting, X. Yang, S. Cheng, and K. Huang, "Hybrid Metaheuristic Algorithms: Past, Present, and Future," In Recent advances in swarm intelligence and evolutionary computation, no. January, pp. 71–83, 2015, doi 10.1007/978-3-319-13826-8.
[44] M. Alardhi and A. W. Labib, "Preventive maintenance scheduling of multi-cogeneration plants using integer programming," Journal of the Operational Research Society, vol. 59, no. 4, pp. 503–509, 2008, doi: 10.1057/palgrave.jors.2602386.
[45] G. A. Bakirtzis, P. N. Biskas, and V. Chatziathanasiou, "Generation expansion planning by MILP considering mid-term scheduling decisions," Electric Power Systems Research, vol. 86, pp. 1–40, 2012, doi: 10.1016/j.epsr.2011.12.008.
[46] N. M. Tabari, A. M. Ranjbar, and N. Sadati, "Promoting the optimal maintenance schedule of generating facilities in open systems," in In Proceedings. International Conference on Power System Technology, 2002, vol. 1, pp. 641–645, doi: 10.1109/ICPST.2002.1053621.
[47] R. Tonić and M. Rakić, "Annual preventive maintenance scheduling for thermal units in an electric power system," Yugoslav Journal of Operations Research, vol. 20, no. 2, pp. 261–273, 2010, doi: 10.2298/YJOR1002261T.
[48] K.-Y. Huang and H.-T. Yang, "Effective algorithm for handling constraints in generator maintenance scheduling," in IEE Proceedings - Generation, Transmission and Distribution, 2002, vol. 149, no. 3, pp. 274–282, doi: 10.1049/ip-gtd:20020214.
[49] I. Kuzle, H. Pandžić, and M. Brezovec, "Hydro Generating Units Maintenance Scheduling Using Benders Decomposition," Technical Gazette, vol. 17, no. 2, pp. 145–152, 2010, doi: 10.1117/12.624407.
[50] K. H. Chung, B. H. Kim, and D. Hur, "Distributed implementation of generation scheduling algorithm on interconnected power systems," Energy Conversion and Management, vol. 52, no. 12, pp. 3457–3464, 2011, doi: 10.1016/j.enconman.2010.10.006.
[51] Muthana, S.A. and Ku-Mahamud, K.R., 2021. Generator Maintenance Scheduling Models for Electrical Power Systems: A Review. Int. J. Electr. Electron. Eng. Telecommun, 10(5), pp.307-318.
[52] M. Buljubasic and H. Gavranovic, "Orchestrating Constrained Programming and Local Search to Solve a Large Scale Energy Management Problem," in Proceedings of the Federated Conference on Computer Science and Information System, 2012, pp. 371–378.
[53] F. Brandt, R. Bauer, M. Völker, and A. Cardeneo, "A constraint programming-based approach to a large-scale energy management problem with varied constraints," Journal of Scheduling, vol. 16, no. 6, pp. 629–648, 2013, doi: 10.1007/s10951-012-0281-1.
[54] S. Kheawhom, "Efficient constraint handling scheme for differential evolutionary algorithm in solving chemical engineering optimization problem," Journal of Industrial and Engineering Chemistry, vol. 16, no. 4, pp. 620–628, 2010, doi: 10.1016/j.jiec.2010.03.004.
[55] H. Sun, Y. Huang, and K. Huang, "A New Algorithm for Power System Scheduling Problems," in Eighth International Conference on Intelligent Systems Design and Applications, 2008, pp. 36–40, doi: 10.1109/ISDA.2008.200.
[56] J. T. Saraiva, M. L. Pereira, V. T. Mendes, and J. C. Sousa, "A Simulated Annealing based approach to solve the generator maintenance scheduling problem," Electric Power Systems Research, vol. 81, no. 7, pp. 1283–1291, 2011, doi: 10.1016/j.epsr.2011.01.013.
[57] J. Sugimoto, A. M. Isa, and R. Yokoyama, "Profit-oriented Thermal Unit Maintenance Scheduling under Competitive Environment," in In 2007 IEEE Power Engineering Society General Meeting, 2007, pp. 1–6.
[58] A. Vlachos, "Rank-Based Ant Colony Algorithm For A Thermal Generator Maintenance Scheduling Problem," WSEAS Transactions on Circuits & Systems, vol. 12, no. 9, pp. 273–285, 2013.
[59] M. Fattahi, M. Mahootchi, H. Mosadegh, and F. Fallahi, "A new approach for maintenance scheduling of generating units in electrical power systems based on their operational hours," Computers & Operations Research, vol. 50, pp. 61–79, 2014.
[60] U. E. Ekpenyong, J. Zhang, and X. X. Centre, "An improved robust model for generator maintenance scheduling," Electric Power Systems Research, vol. 92, pp. 29–36, 2012.
[61] Ismail, F.B., Randhawa, G.S., Al-Bazi, A. and Alkahtani, A.A., 2023. Intelligent Optimization Systems for Maintenance Scheduling of Power Plant Generators. Information Sciences Letters, 12(3), pp.1319-1332.
[62] N. Kumarappan and P. Suriya, "Hybrid GA / SA Based Generation Maintenance Scheduling with Line Flow Constraints in Power Market," in In 2010 Joint International Conference on Power Electronics, Drives, and Energy Systems & 2010 Power India, 2010, pp. 1–8.
[63] T. Jayabarathi, S. Chalasani, and Z. A. Shaik, "Hybrid Differential Evolution and Particle Swarm Optimization Based Solutions to Short Term Hydro Thermal Scheduling," WSEAS TRANSACTIONS on POWER SYSTEMS, vol. 2, no. 11, pp. 245–254, 2007.
[64] E. Reihani, M. O. Buygi, and M. Banejad, "Generation Maintenance Scheduling Using Hybrid Evolutionary Approach," in The International Conference on Electrical Engineering 2008, 2008, pp. 1–7.

[65] E. B. Schlünz and J. H. van Vuuren, "the application of a computerized decision support system for generator maintenance scheduling: a south african case study," south African Journal of Industrial Engineering, vol. 23, no. 2, pp. 169–179, 2012.
[66] G. G. Samuel and C. C. A. Rajan, "Hybrid Particle Swarm Optimization – Genetic Algorithm and Particle Swarm Optimization – Evolutionary Programming for Long-term Generation Maintenance Scheduling," in International Conference on Renewable Energy and Sustainable Energy [ICRESE'13], 2013, pp. 227–232.
[67] D. K. Mohanta, P. K. Sadhu, and R. Chakrabarti, "Fuzzy reliability evaluation of captive power plant maintenance scheduling incorporating uncertain forced outage rate and load representation," Electric Power Systems Research, vol. 72, no. 1, pp. 73–84, 2004, doi: 10.1016/j.epsr.2004.04.001.
[68] D. K. Mohanta, P. K. Sadhu, and R. Chakrabarti, "Fuzzy Markov model for determination of fuzzy state probabilities of generating units including the effect of maintenance scheduling," IEEE Transactions on Power Systems, vol. 20, no. 4, pp. 2117–2124, 2005, doi: 10.1109/TPWRS.2005.857932.
[69] R.-C. Leou and S.-A. Yih, "A flexible unit maintenance scheduling using fuzzy 0-1 integer programming," in In 2000 Power Engineering Society Summer Meeting (Cat. No. 00CH37134), 2000, vol. 4, pp. 2551–2555.
[70] Liu, S., Liu, J., Ye, W., Yang, N., Zhang, G., Zhong, H., Kang, C., Jiang, Q., Song, X., Di, F. and Gao, Y., 2023. Real-time scheduling of renewable power systems through planning-based reinforcement learning. arXiv preprint arXiv:2303.05205.
[71] Pinninti, S. and Sura, S.R., 2023. Renewables based dynamic cost-effective optimal scheduling of distributed generators using teaching–learning-based optimization. International Journal of System Assurance Engineering and Management, pp.1-21.