Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 5 (2023)

Genetic Algorithms: Review of Recent Variants and Applications

Esra'a Alkafaween[1] , Ahmad B. Hassanat[1], Ehab Essa[2], Samir Elmougy[2]
[1]*Computer Science Department, Mutah University, Karak 61711, Jordan*
[2]*Department of computer science, Mansoura University, Mansoura, Egypt*

Abstract: - Genetic Algorithm (GA) is a meta-heuristic inspired by evolution, that belongs to the broad family of evolutionary algorithms. These algorithms are widely employed to provide high-quality solutions to optimization problems by concentrating on evolutionary biology operators (biological-inspired operators) such as: mutations, crossover, and selection. This research aims to provide a comprehensive overview of GAs. The various research areas involved in GAs are covered, such as: the field of genetic operators, fitness function and hybrid algorithms, in addition to discuss GA models. Also, some other information on diversity maintenance methods is given, to allow readers to widen their knowledge in this field.

Keywords*: Evolutionary Algorithms, genetic algorithms, Diversity, multipopulation, premature convergence, parallel GA.*

1. Introduction

In recent years, real-world applications have become significantly more complex. Bioinformatics, Robotics, data mining, operations research, machine learning, decision making, and many other topics are extremely complicated and difficult to solve. Therefore, the term Evolutionary Algorithms (EAs) appeared to deal with such complex problems [1] [2]. EAs are a group of algorithms that were originally created to handle combinatorial optimization problems [3]. They try to solve optimization problems by mimicking Darwinian evolution process [4] [5]. They are considered as a branch of Evolutionary Computation (EC). EC is a subfield of Computational Intelligence that is a special branch of Artificial Intelligence (AI) [6]. Figure 1 depicts different approaches to optimization strategies, with a focus on evolutionary approaches.

EAs are population-based metaheuristic optimization algorithms that mimic the theory of natural evolution [8], [9], [10], [11] and [12]. Using the survival of the fittest concept, evolutionary algorithms work on a set of possible solutions to produce the best solution or the closest to the best [13]. The process of choosing individuals based on their degree of fitness in the problem domain and breeding them using operators (such as crossover and mutation) taken from natural genetics, produces a new set of approximations at each generation [3], [6]. Thus, this process results in the evolution of groups of individuals that are more adapted to their environment than those from prior generations [14]. Figure 2 shows the diagrammatic view of simple EA.

In general, The EAs family includes the following main algorithms: Genetic Algorithm (GA) [16], Evolution Strategy (ES) [17], Evolutionary Programming (EP) [18], Genetic Programming (GP) [19], and Differential Evolution (DE) [20]. Each of these approaches has several variations, and each is utilized in a wide range of industrial applications [7].

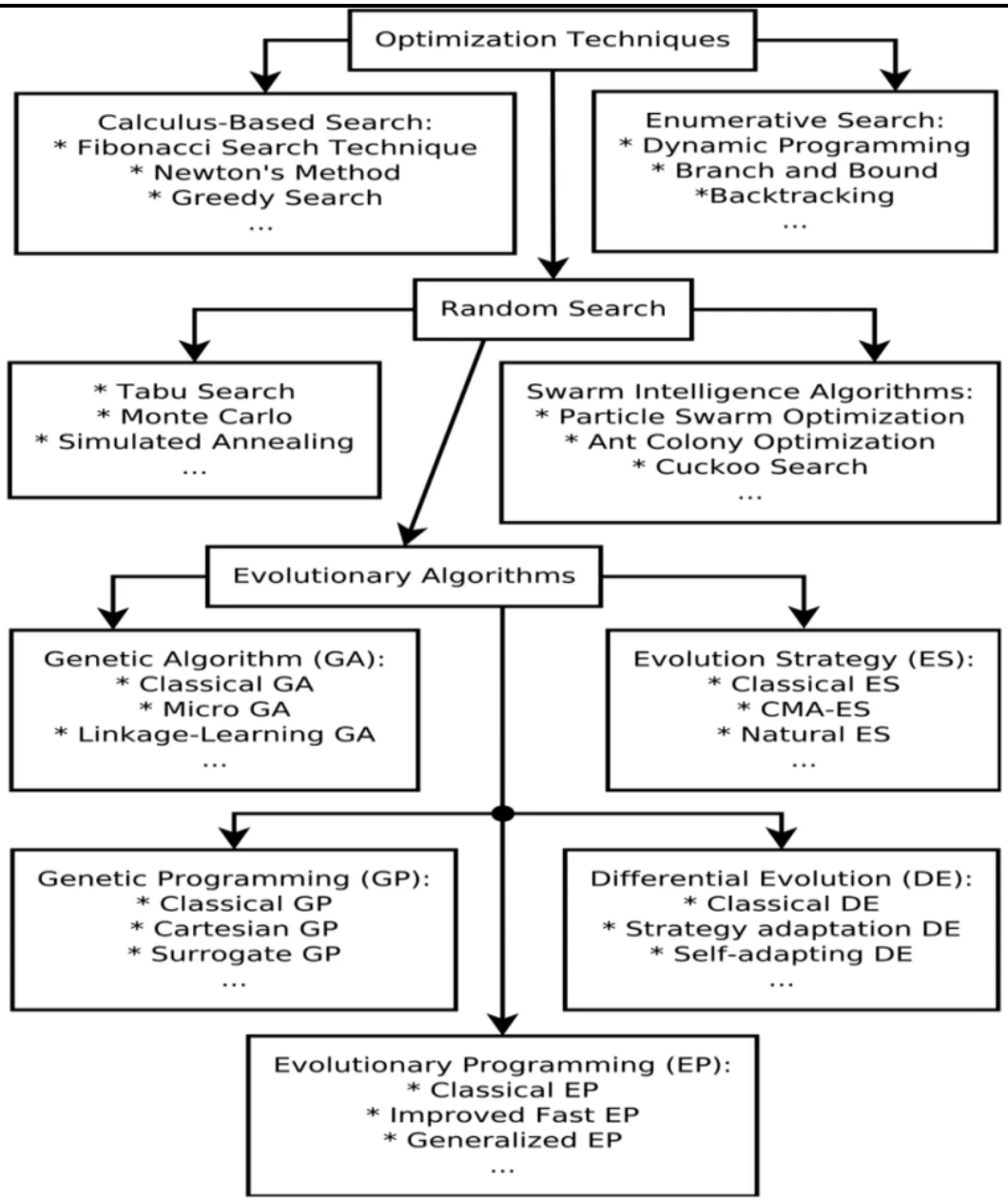

Figure 1 Classification of nature-inspired approaches [7]

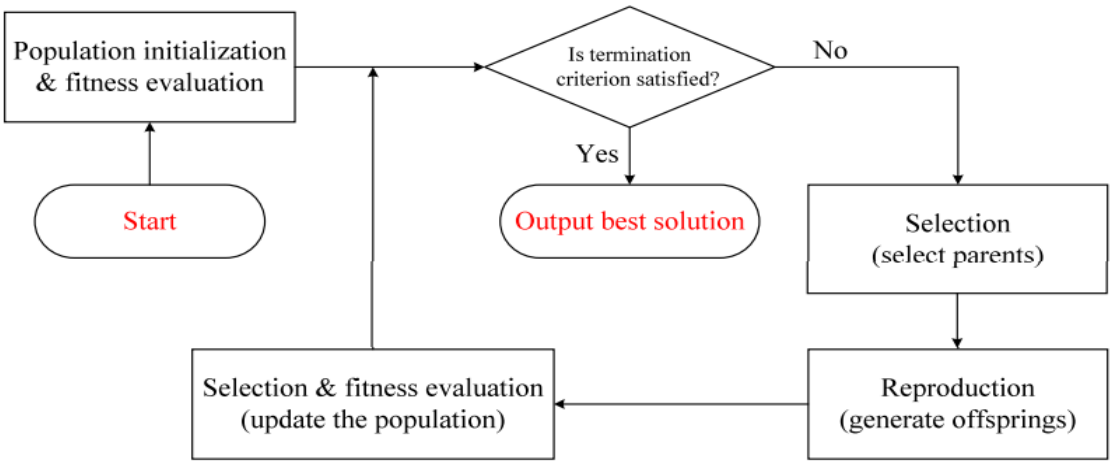

Figure 2 the general framework of the evolutionary algorithms [15]

1.2 Genetic Algorithms

GA is the most common type of EA [21], because it demonstrates the most accurate mapping of the natural evolution process on a computer [2]. It was introduced in the early 1970's in the University of Michigan by John Holland [22]. It is a stochastic optimization technique inspired by the theory of natural selection [23]. It is a population-based search method that uses the principle of survival of the fittest, that generates fresh sample points in a search space by using selection and recombination (crossover) operators [24] [25] [26].

GA's popularity has grown and spread among researchers in various domains [27], since it has demonstrated its ability to solve a wide range of problems [28], and it is therefore regarded as an optimization tool [29] [30] and [26]; some examples of these domains are: robotics [31], software engineering [32], computer networks [33], speech recognition [34], Natural language processing [35], image processing [36], etc.

GAs, in contrast to local search techniques, are classified as global search heuristics and are based on a series of independent computations managed by a probabilistic approach [37]. GA begins with a set of solutions known as the population. A chromosome represents solutions. The population size remains constant from generation to generation. At the end of each generation, each chromosome's fitness is assessed, and chromosomes for the following generation are probabilistically picked based on their fitness ratings. Some of the chosen chromosomes mate at random and generate children (offspring). Crossover and mutation occur at random while creating offspring. The average fitness value of the new generation chromosomes may be higher compared to the fitness of the old generation, because there is a high probability of selecting chromosomes with high fitness values. The evolution procedure is repeated until the final condition is met [24]and [38].

2. Canonical GA (CGA)

GA developed by John Holland in 1965 is referred regarded as the classic (Canonical) GA. Canonical GA (CGA) is depicted in Figure (3). It uses a binary/bit-string format to encode the genome, uses roulette-wheel proportional for selection, and relies on uniform mutation and on a single point crossover [26].

GA deals with a large number of potential solutions (population), in which each solution is represented by a different chromosome. The initial step is to encode all feasible solutions onto a chromosome, which is referred to as "encoding" or "representation" of problem solutions (e.g. binary encoding). The next steps are described as follows [39]:

1. Initialization: Generating a random population [40], in which each member of this population in the CGA will be a binary string of length L that matches to the problem [26].
2. Evaluation: To distinguish good solutions from bad ones, each chromosome's fitness is calculated. Fitness is the term used in GAs to describe the measurement of how near an individual is to the goal [41].

3. New Generation: Repeating the steps from 6 to 8 to generate a new population until the new population is finished.

4. Selection: According to the fitness value, two chromosomes are chosen from the population, the better ones are more often picked for reproduction than the bad ones; this is done at random, with a probability based on the individuals' relative fitness. In CGA, the roulette wheel Selection is used.

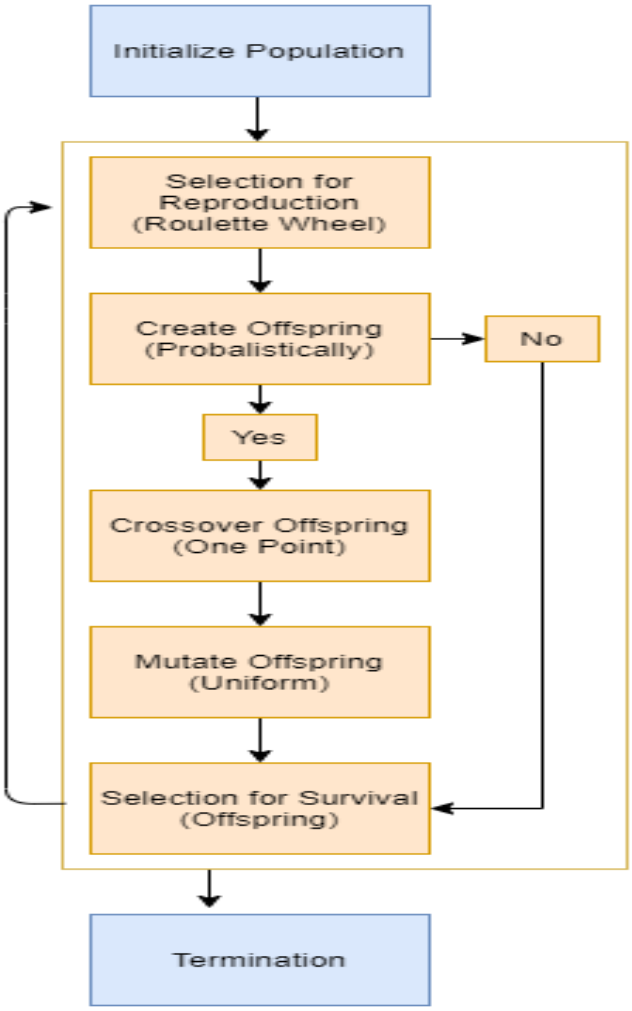

Figure 3 Flowchart of Canonical Genetic Algorithm [42]

5. Reproduction: The process may employ both crossover and mutation to create new chromosomes:

- Crossover: Parts of two or more parental solutions are combined to create new, maybe superior solutions through recombination (i.e. offspring). This can be accomplished in a variety of ways (some of which are discussed in the next section). In CGA, the single-point crossover operator with crossover probability is used.
- Mutation: Mutation modifies a solution locally, but randomly, and generally it entails one or more modifications to an individual's attribute or traits. Again, there are a lot of different mutations in which some of them are discussed in the next section. In CGA, the uniform mutation operator is used to generate offspring with a mutation probability.

6. Regeneration (Replacement): Individuals from the previous population are replaced by new ones in the last stage.

7. Steps 2–6 should be repeated until a termination condition is fulfilled.

8. Criteria for Stopping: Stop and return the best solution in the current population if the end condition is met. Several terminating conditions have been imposed [43], such as [25] [43]:

- Achieving the maximum number of generations
- Improving fitness continues to fall short of the threshold value and the opportunity to affect change in future generations has limited.
- Implying that the likelihood of population variety has increased.

3. GA Structure Modification

A standard GA can solve a complex problem that traditional gradient-based or hill-climbing approaches might struggle to solve. On the other hand, a standard GA, has several flaws. The structure of GA must be changed to meet the need when the problem grows more difficult, multitasking, and contradictory. There are also many aspects of change that must be introduced on chromosomes, operators, and initial populations [44], [45].

1. Chromosome Encoding (Representation)

The first stage in implementing GA is to establish a link between the actual world and the GA's universe. This is referred to as "encoding". Encoding is mostly determined by the problem to be addressed, as there are several encoding techniques available. As a result, a suitable methodology must be chosen. The following are some examples of encoding techniques:

- Binary encoding [16]: In this method of encoding, a chromosome is represented by a binary string (e.g. Knapsack problem).
- Value encoding: A series of values is used to represent each chromosome. These values might be real number or character, etc. (e.g. a neural network) [46].
- Permutation encoding: This form of chromosome represents a place in a sequence (e.g., TSP) [47].
- Tree encoding: A tree of functions or instructions represents the chromosome (e.g. Floor planning) [48].

2. Initial Population

The first phase of every GA application is the population seeding phase. As input for GA, it produces a population of possible solutions or individuals at random or through heuristic initialization. Because the quality of individual solutions created in the initial population stage is significant in determining the quality of the ultimate optimal solution, population initialization is an important phase in GA [49]. Random initialization is the most frequent approach for generating the initial population in GA [50] [51] and [52]. Random approach has a low fitness solution, which reduces the chances of discovering an optimal or near-optimal solution; it also takes a considerable search time if information is lacking. In a large search space, on the other hand, if prior heuristic knowledge about the best solution is provided, it may simply build the initial population and identify high-quality solution regions. The use of heuristic approaches to generate initial population seeding results in a high-quality population, allowing GAs to identify better solutions quicker. However, it is possible that it will end up with a narrow search space and will never be able to find globally optimum solution [53].

Since the inception of GA ideas, a variety of initialization strategies have been introduced, such as: Gene bank initialization technique [54], Nearest Neighbor Initialization Technique [55], K-means Initial Population (KIP) [56], Initialization Mechanism Based on Regression Techniques [53] [51], and Equi-begin Vari-diversity (EV) [56].

3. Selection Mechanism

The selection operator aims to leverage the best attributes of excellent candidate solutions in order to enhance these solutions across generations, which should, in theory, help GA to converge to an acceptable and adequate result of the optimization issue at hand [57]. The most crucial aspect that can impact a GA's performance is the selection operator [58] [59].

Several selection methods had been proposed in the literature, including Roulette Wheel Selection, Stochastic Universal Sampling, Tournament Selection, and Boltzmann Selection, among others. Despite decades of research,

no universal criteria or theoretical backing exist for picking an appropriate selection procedure for each problem [60] [61]. A summary of each mechanism is presented below:

- Fitness Proportionate Selection: Fitness proportional selection was the first type of selection presented when genetic algorithms were initially established. Because of the likeness of selection to an actual roulette wheel; it is also known as Roulette Wheel Selection (RWS) [62]. It uses the fitness function to assign a fitness score to each chromosome (individual) in the population. Fitness level is utilized to correlate a likelihood of selection with each chromosome, and it is used to determine which solution is best [63]. Due to the possibility of the presence of a dominating person who always wins the competition and is picked as a parent, a well-known disadvantage of this approach is the potential of early convergence of GA to a local optimum [57].
- Stochastic Universal Sampling (SUS): This mechanism is based on the kind of fitness proportionate selection; it chooses solutions at equally spaced intervals using a single random value to sample all of them. One of its downsides of this mechanism is the premature convergence [24].
- Linear Rank Selection (LRS): LRS is one of the most widely used selection strategies, as well as a version of RWS that attempts to address the problem of premature convergence. It is based on an individual's rank rather than their fitness [61]. Low convergence sorting required, and computationally expensive, are some of its disadvantages [24].
- Tournament Selection (TOS): Due to its effectiveness and ease of implementation, this sort of selection approach is arguably the most prevalent in GAs [64]. In tournament selection, n people are chosen at random from a bigger population, and the chosen individuals compete against one another. The person who achieves the best level of fitness wins and becomes a member of the next generation population. Tournament selection also allows all individuals a chance to be chosen, preserving diversity, albeit maintaining diversity may slow convergence pace [65].
- Truncation Selection (TRS): This approach is a relatively basic technique that ranks the possible solutions of population in order of fitness. Except for a very large population, it is less commonly employed in practice than other strategies [61]. The population sorting determines the time complexity of the truncation selection. Using a merge sort, for example, assures that the time complexity is O (n log n). While the truncation sort is the fastest of the choices presented, it has the disadvantage of excluding the greatest information variety in a given evolutionary set. Because only the best options are ever explored, the suggested ultimate solutions may become trapped at a local maximum of being a good, but not the best solution [66].
- Boltzmann Selection: This type is based on Monte Carlo Simulation's entropy and sampling methodologies. It aids in the solution of the premature convergence problem [67]. The chances of picking the best string are really good, and it takes very little time to do so. However, there is a risk of data loss [24].
- Elitism Selection: The initial best chromosomes or a few best chromosomes could be replicated to the new population via elitist selection. To begin, the chromosomes might be arranged in decreasing order. After that, every two chromosomes in the arrange set are selected. We don't need to make any changes since we can pass on the greatest individuals to the next generation [68].

4. Crossover (Recombination) Operator

By exchanging parts of the parent genes, two parents (chromosomes) can produce a new child. It's more probable that the new offspring (children) will inherit good characteristics from their parents and, as a result, do better than their forefathers [69] [70].

Many different types of crossover had been produced throughout the years. The type of crossover we choose is mostly determined by the type of encoding employed; this can be difficult at times, but it usually increases the genetic algorithm's performance [71]. The first type was one-point crossover and expanded into a variety of approaches to handle a variety of situations such as Ring crossover [72], Uniform crossover [73], Heuristic crossover [74], Multi-point crossover [75], Arithmetic crossover, and for the order-based problem, Cycle crossover (CX) [76], the Partially Matched crossover (PMX) [77], Order crossover (OX) [30], Collision Crossover, and Select the best crossover (SBC) [69].

5. Mutation Operator

It is the process of changing or swapping certain genes inside a single chromosome in order to build chromosomes that supply new solutions for the following generation [70]. Because of its influence on the investigation of global optima, mutation is one of the most critical stages of GA [78] [79], [80] It avoids the convergence of the local optimum by exploring fresh locations in the search space [81]. Holland invented classical mutation (Bit-flip Mutation) and because mutation is a critical process in the search process, several mutations have been developed in the literature such as Exchange Mutation, Gaussian Mutation, [82], Uniform Mutation and Creep Mutation [83], Inversion Mutation [84], Displacement Mutation [85], Worst gene with worst gene mutation [86], and IRGIBNNM [87].

6. Replacement (Reinsertion)

We must introduce the new offspring solutions into the parental population after they have been formed through crossover and mutation. There are several approaches may be used. The parent chromosomes were previously chosen based on their fitness, so it is expected that the offspring are among the fittest in the population, and that the population will progressively, on average, raise its fitness [39] [88]. The following are some of the most frequent replacement strategies:

- Delete All: This approach removes all members of the present population and replaces them with the same number of newly generated chromosomes. Due to its relative ease of implementation, this is likely the most frequent strategy and will be the technique of choice for most people.
- Steady-State: This approach replaces n existing members with n new members by deleting n old members. This deletion technique's parameter, n, is the number of items to remove and replace at any given moment. Another factor to consider while using this strategy is determining which members of the present population to remove [39] [89], are the worst population eliminated? or randomly selected, or eliminate the chromosomes that were used as parents? This is, again, a technical parameter.
- Steady-State-no-Duplicates: This approach verifies that no duplicate chromosomes are introduced to the population, much like the steady-state strategy; this increases the processing burden but allows more of the search space to be investigated [39].
- Expansion Sampling: It combines the new individuals with the previous generation and selects the fittest half-individuals for the following generation. The traditional GA places new individuals straight into the next generation, denying older, better individuals who have been crossed or mutated access to future generations, slowing the population's convergence rate. The large-scale optimum sampling ensures that the better individuals make it into the following generation, speeding up convergence [90].

4. GA Control Parameter

To manage their evolutionary search for a solution to their assigned issues, GAs employ a number of parameters. The rate of crossover and the rate of mutation are two examples. The maximum number of generations, the total number of individuals in the population, and so forth. When it comes to selecting proper values for these parameters, there are no hard and fast rules. A set of control parameters that is optimum or near-optimal for genetic algorithm application does not apply to all circumstances. Choosing settings for control parameters is sometimes a trial-and-error process. Hand-tuning each of the control settings one at a time is a typical procedure. This may be a time-consuming and exhausting process [91]. The basic parameters are as follows:

- Population Size: The population size is an important parameter that has a direct impact on the capacity to find the best solution in the search space. In classical GA, the user establishes a fixed population size that remains constant throughout the iterative process. Many studies have shown that having a big population increases the accuracy of finding the best solution. However, if the search space is limited, a large population size is not a good option.. As a result, the ideal population size has been found, however the size is set. Despite the fact that the population size is ideal, the fixed population causes temporal complexity and complicates the search by increasing the number of generations required to converge. As a result, the size of the population size must be constantly

varied throughout the GA development of new solutions [92] [93]. Many researchers have looked at the impact of selecting a suitable initial population size on GA performance as in [94], [95], [96], [97], [98], and [99] .

- Probability of Crossover (PC): Crossover probability is a crucial factor in generating new children from a pair of chromosomes from two different parents. There is no crossover if the crossover probability is zero; offspring have the same chromosomes as their parents in the preceding population. Crossover probability of 100% indicates that crossover is responsible for all offspring. In the hopes that the new chromosome population would contain both good sections of old chromosomes and maybe new chromosomes that will converge the solution, a tradeoff between "0" and "100%" likelihood of crossover" would be a preferable tradeoff to evaluate. It's a good thing that some chromosomes from the previous population's generation make it to the next [100]. If the PC value is too high, new people are quickly created. Genetic models might be destroyed at the same time. The search procedure will be excessively slow if the value of pc is too tiny [101].

The fixed crossover rate operator is used by traditional genetic algorithms. Because the crossover rate of individuals is the same, all of the contemporary individuals in the cross-operation will be retained at the same probability, resulting in the current better individuals being selected several times at the choice operation of the next round, and the relatively poor individuals in the current generation being eliminated, causing the population to rapidly evolve in the direction of the current optimal individual. If the current optimal individual is a local optimum, the entire algorithm can quickly go into local optimum [90].

- Probability of Mutation (Pm): To avoid the local optimal solution, the mutation operator's probability should be between 0 and 100, and a tradeoff probability will give good new offspring to reach the best solution [100]. If the value of Pm is too high, genetic algorithms can easily degenerate into a pure random search approach. However, if the value of Pm is too low, it is impossible to develop new individuals [101]. A fixed mutation rate is used in the classic genetic algorithm's calculation. As a result of the low mutation probability when the algorithm is in local optimum, it is difficult to jump out of it; however, in certain cases, even if they jumped out, they would be ineffective since they would not be picked in the following choice procedure [90].Several researches had arisen with the notion of dynamic PC and Pm to prevent this scenario and maintain population diversity, such as: [102], [103], [104], and [105].

5. Premature Convergence

GA's premature convergence occurs when the genes of a few highly ranked individuals suddenly come to dominate the population, forcing it to converge to a local optimum. The loss of variety within the population is usually the cause of early convergence. The selection pressure, the schemata dispersion related to crossover operators, and a bad evolution parameter setup can all contribute to this loss. When the population of a GA reaches a suboptimal state, the genetic operators are no longer able to create children that perform better than their grandparents. To avoid early convergence, it is critical to sustain population variety throughout evolution in a GA [106] [107]. Premature convergence and slow convergent speed are common downsides of GA [108]. Although GA has not attained a global or satisfying optimum, it is unable to create children that exceed their parents due to premature convergence. It is difficult for GA to eliminate a local optimum and obtain a global optimum when premature convergence occurs [109] [110].

Many studies had evolved to avoid premature convergence, such as: Xin et al. established the notion of multi-domain inversion to expand the number of offspring for the objective of improving local search capability and raising the likelihood of producing great individuals [111]. Peng et al. [112] presented Low Visit Cost Crossover (LVC) method, which picks the genetic pieces based on nodes and anchor points. This method ensures that the population is distributed more evenly. In [108], researcher introduced the "diversity-guided genetic algorithm-convolutional neural network (DGGA-CNN)", which employs adaptive parameter management and random injection to simplify the search process through exploration and exploitation while maintaining population variety. A GA-based Edge Selection approach (ESGA) is suggested in [113]. The edge selection-based technique may create viable solutions during initialization and ensures that each feasible solution is generated with a reasonable probability, which improves the efficiency of GAs convergence.

6. Diversity

Diversity refers to how chromosomes in a population differ from one another. We are continuously looking for diversity, and its lack is the main reason for the population to reach local minima [114]. Members of the gene pool are too "similar" or lose diversity, which is the obvious explanation for early convergence [115]. However, the phrase population diversity had been used qualitatively in several articles to explore premature convergence. This insight implies that one way to avoid convergence is to ensure that various members of the gene pool are distinct. Because each structure is encoded as a bit string, it is sufficient to ensure that when a new structure is put to the pool, it varies from another structure by at least one bit. If the new individual is similar to another individual of the gene pool, modify one bit at a time and continue until the outcome is distinct from every other individual of the pool [116].

Diversity measures are used to offer information about the search state, such as exploring or exploiting [2]. Measuring diversity can therefore provide information into an algorithm's behavior at a given time step [117].

There are several methods for measuring the genetic diversity, such as those found in: [118] [117] [119] and [120] and many approaches have arisen in recent years to enhance and sustain population diversity and thereby minimize premature convergence. In addition to dealing with the multi-modal function [81], this would aid progress by providing global exploration support and gaining access to multiple global and local optima [121] , [122]. Some of the used approaches are listed below.

1. Multi-Population Genetic Algorithm (MPGA):

MPGA algorithm is also known as the "island model", the basis of which is to divide the population into sub-populations, where each island (sub-population) is more likely to take a distinct search path [123]. The migratory mechanism exchanges good individuals across subpopulations [122], while the crossover operator generates new people. This enables for the discovery of hitherto unknown places. The migration rate is the number of individuals to be swapped between segments of the population (sub-population), and it permits control of the level of diversity within the sub-population [122]. Another variable that encourages the subpopulation diversity is the migration interval: this variable affects the number of times a subpopulation migrates.

The speciation method, which mixes the chromosomes based on genetic similarities, can be used to integrate a multi-population. The Euclidean distance, for example, is used to quantify similarity. Numerous peaks are formed in this process; hence there are multiple solutions to the problem rather than just one.

2. Primal Dual GA (PDGA) [124]

This approach works with a pair of chromosomes known as "primal-dual." Chromosomes registered in the GA population are referred to as "primal," while other chromosomes with the greatest distance (in genotype) in a given distance space for the primal chromosomes are referred to as "dual," and the modification task from primal to dual using a distance measure (e.g. Hamming distance) is referred to as "primal-dual mapping." The algorithm selects chromosomes with lower fitness to offer them a chance to go to their better dual. This method improves the search's efficiency.

There are also various approaches that promote population diversity, such as Crowding, Elitist, Dual Population GA (DPGA) [125], Multi-Objective Evolutionary Algorithm (MOEA), Diversity Control Oriented GA (DCGA), Injection Strategy, and Restricted Mating.

3. Nitching Method:

Niching approaches had been devised to reduce the influence of genetic drift caused by the selection operator in classical GA [126], allowing for the simultaneous examination of numerous solutions in the population. In natural systems, animals compete and thrive in a variety of ways (for example, via grazing and hunting), and various species develop to fulfil each role. A niche may be thought of as an organism task that allows organisms to exist in their environment. A species is defined as a group of comparable organisms having similar characteristics. Physical resources are limited in each niche and must be shared by the inhabitants of that niche. A niche is frequently referred to as a domain optimum, with the fitness indicating the niche's resources. In terms of similarity

metrics, species may be characterized as comparable individuals [127]. Examples of such applications are multi-objective optimization, multi- modal functions, and classification.

7. Enhancements of Genetic Algorithms

GAs had been improved over the years from what was popular during the Holland era [128], to meet some of the demands, such as time-dependent and optimizing multi-modal problems. As a result, numerous other extensions of the conventional GA have emerged, such as hybrid GAs and parallel GAs. There are a vast number of studies in the literature that address GA enhancements, the ultimate objective is to boost the efficiency of the GA by increasing the population diversity and reducing the premature convergence.

7.1 Parallel GA (PGA)

Sequential GAs had been proven to be quite effective in a variety of applications and areas. However, there are a few problems with their applications that can all be solved with some type of PGA [129] and [130] such as:

- For some problems, a large population is necessary, and the amount of memory necessary to keep each individual may be significant. In some circumstances, this makes running an application efficiently on a single computer impractical, forcing the use of a parallel version of GA.
- The process of evaluating individual's fitness normally takes a long period. In complicated domains, calculation periods of more than one CPU year have been recorded in the literature for a single run. It leaps to reason that parallel processing is the only viable technique to deliver this CPU power.
- Sequential GAs may become stuck in a sub-optimal part of the search space, preventing them from finding better solutions. PGAs can explore many subspaces of the search space in concurrently, reducing the likelihood of being stuck by low-quality subspaces.

The basic concept underlying the most parallel programs is to break a job into pieces and to solve each piece utilizing several processors at the same time. This divide-and-conquer strategy may be used to GAs in a variety of ways, and there are several instances of effective parallel implementations in the literature. Some parallelization approaches work with a single population, while some others split the population into numerous small subpopulations [131]. The following is the three main categories of Parallel GAs, as shown in Figure 4 – Figure 6 [132] [133] [131].

1. Global Single-Population Master-Slave GAs

This type, that is also known as Global Parallelization GA (GPGA) or distributed fitness function, has the population as the master, while the slaves undertake the evaluating individuals, applying mutations, and occasionally the crossover process. There is no requirement for touch when individuals are assessed (the procedure is done in parallel), but there is contact when the slave receives individuals or returns the fitness values upon completion of the evaluation.

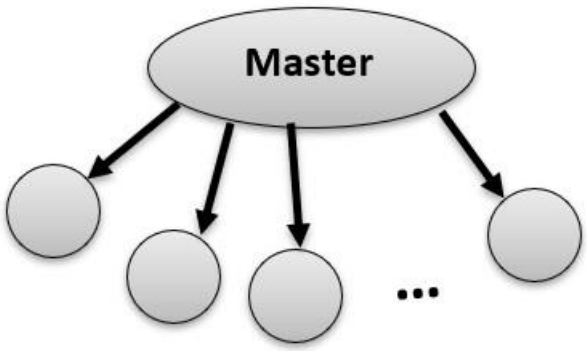

Figure 4 global parallelization

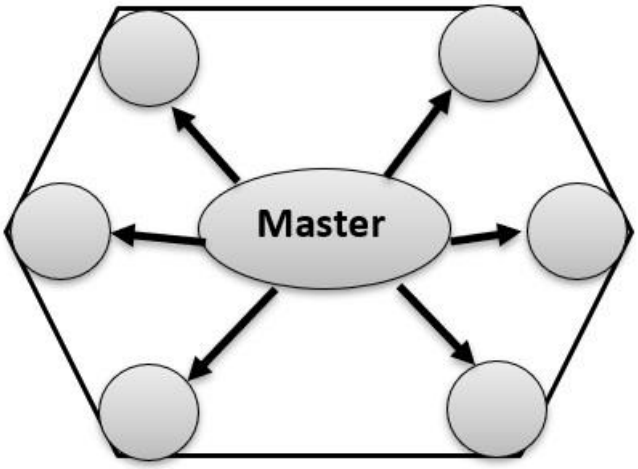

Figure 5 Coarse grain PGA

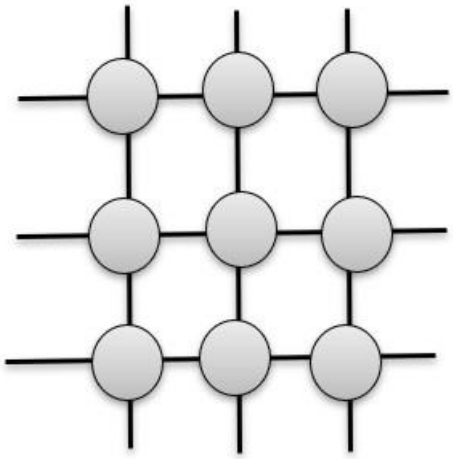

Figure 6 Fine-grain FGPGA

2. Single-Population Fine-Grained (FGPGA)

Massively PGA (MPGA) or Diffusion GA are terms for fine-grained GA. Individuals in FGPGA can only mate with their neighbors and overlaps across neighbourhoods contribute for good individual spreads over the whole population. The function considers the form and the size of the neighbourhood as variables in determining the amount of overlap [134].

3. Multiple-Population Coarse-Grained GAs

This type of Parallel GAs is known by a variety of names, including distributed (DGA) or multiple-demes GA, multiple-population, and island parallel GA. The goal is to split the population into demes (small groups of people), each of which works independently. The propagation of good individuals is induced by migration across sub-populations, which improves the solution. There are many other types of migration topologies, such as injection migration, ring migration and neighborhood migration. The migration rate, sub-population size, migration interval and number of sub-populations are all variables that might influence the search path and efficiency. One challenge for this method is the chance that new individuals would be ineffectual, implying that the resultant genetic material will be incompatible. The explanation for incompatibility in practice is that when two subpopulations of excellent people are mixed, they generate terrible people [135].

7.2 Hybrid Genetic Algorithms (HGAs)

As with any hybrid system, hybrid GAs are built on a complementary perspective of search approaches. Genetic and other search strategies can be viewed as complimentary tools that can be combined to achieve an optimization target. A GA integrates one or more approaches to increase the performance of the genetic search in these hybrids [136]. In other words, a hybrid GA is a GA that has been combined with a local search strategy. The following are the steps in a basic HGA technique [137]:

1. As the existing parent population, produce the starting population at random.
2. In the initial population, determine the fitness function for each chromosome.
3. Create a population of offspring via GA operators (crossover, mutation, selection).
4. Evaluate the fitness function of each individual.
5. Conduct a local search on each offspring, and of each new location, assessing the fitness, and replace the offspring if a better solution exists nearby.
6. Replacement.
7. When a stopping requirement is met, the procedure is terminated. If not, go to Step 3.

Without Step 5, HGA is nothing more than a GA. As a result, HGAs have all of the characteristics that GAs have. HGAs, like GAs, are a vast family of algorithms with similar fundamental structures but differ in various techniques such as stopping criteria, operators that regulate the search process.

Although evolutionary algorithms may quickly detect the region where the global optimum occurs, it takes a considerable time to determine the exact local optimum in the zone of convergence [138]. GA combined with a local search approach can expedite the search for the precise global optimum. Performing a local search to solutions directed by a GA to the most promising location in such a hybrid can expedite convergence to the global optimum. The time required to achieve the global optimum can be lowered if local search techniques and local expertise are employed to speed discovering the most likely search region as well as locating the global optimum beginning within its area of attraction. Local search on the population of a GA can introduce the diversity and help to resist genetic drift. It allows for the equitable representation of diverse search regions in order to combat premature convergence. Using a local search algorithm also adds an explicit refining operator, that can yield high-quality results [136]. There are many applications that have adopted the HGA; there are many applications that have adopted the hybrid, such as Feature Selection [139], Routing problem [140], heart disease diagnosis [141] , and Robotics [142] .

7.3 Multiobjective Genetic Algorithms (MOGAs)

Genetic algorithms have primarily been used to solve issues with a single purpose. However, many real-life issues have numerous objective functions. To be dealt by a single-objective genetic algorithm, these objective functions need be integrated into a scalar fitness function. The path of search in GA is determined in the multi-dimensional goal space if a consistent weight is applied to each of numerous objective functions for merging them [143].

MOGAs are a variation of the regular GA [144]. It is a guided random search approach that is one of several engineering optimization strategies. It can solve Multi-objective Optimization Problems (MOPs) and explore different parts of the solution space. As a result, a diversified collection of solutions with more parameters that can be improved at the same time may be found [145]. The Pareto fronts are used to show MOGA solutions [146]. A non-dominated solutions frontier Pareto optimum set is a collection of solutions. The Pareto front refers to the values of the associated objective function in the objective space when the Pareto optimal set is used, simulated annealing, random searches, dynamic programming, and gradient techniques are traditional ways for addressing multi-objective problems, but newer heuristic methods include cognitive paradigms such as simulated annealing, Artificial Neural Networks (ANNs), and Lagrangian approaches. Some of these strategies are effective in locating the best solution, but they have a propensity to take longer to converge, necessitating a significant amount of processing time. As a result, MOGA technique, which is based on the natural biological assessment concept, will be employed to address this type of issue [147]. The following are some Evolutionary Optimization Methods for Solving MOPs [148]:

- Vector Evaluated GA (VEGA)
- Multiple Objective GA (MOGA)

- Micro GA-MOEA (μGA,μGA2)
- Strength Pareto Evolutionary Algorithm (SPEA, SPEA2)
- Weight Based GA (WBGA)
- Pareto Archived Evolution Strategy (PAES)
- Multi-Objective Messy GA (MOMGA-I, II, III)
- Pareto Enveloped Based Selection Algorithm (PESA, PESA II).

MOGA has been used in many real applications, including Auto pilot controller design, Road train design, analog filter tuning, Dose optimization in brachytherapy, and Electrostatic micro motor design.

8. Genetic Algorithm Benefits and Drawbacks

After describing the GA alteration, some insight into why it is becoming more popular should be provided. GA is appealing for a variety of reasons, such as:

- It's a simple technique that requires very little (if any) mathematics [71].
- It's simple to integrate with current simulations [149].
- It can solve multimodal [150], noncontinuous [151], or even NP-complete problems [152], and nondifferentiable problems [153].
- Parallelism [154].
- It can handle Multiobjective problem [155].
- They are immune to being caught in local optima [24].

In fact, there are some tasks that GA is unable to complete or find hard. Some of these limitations are:

- Premature convergence [156].
- The issue of determining fitness function is a difficult one [157] [158].
- Definition of the problem's representation [159] [158].
- The difficulty in determining numerous parameters, such as: mutation rate, cross-over rate, population size and selection technique [160].

9. Conclusion

This study aimed to give a systematic and explained view of GAs. The topic of GA and its variations has been addressed. GA is a basic algorithm that may be easily implemented. It may be used to solve a broad range of problems, including stochastic programming, unconstrained and restricted optimization problems, combinatorial optimization problems, and nonlinear programming.

GA is shown to converge towards the optimal or near to the optimal solution, through selection, crossover, and mutation processes across consecutive generations. Because GAs merges direction and chance in the search in an efficient and effective manner, this basic procedure should generate a quick, useful, and resilient approach. GAs combines the excellent information buried in one solution with good information from another to develop new solutions with good information acquired from both parents, always (hopefully) leading to optimality.

GAs face significant challenges, limiting their applications. GAs was criticized for their excessive computational cost. It is generally computationally expensive to run, which means that it may have to run for a long time to provide good results, and employing a larger population or investigating a much more complex problem slows it down significantly. There are also issues with setting parameters, and how it influences the algorithm's results, the choice of the fit function, the representation of the entities and population seeding are all crucial to producing relevant, meaningful, and satisfying results. On the other hand, several techniques had been proposed with the purpose of overcoming these problems, and research is continuing, and in a remarkable development.

The ability of GA to simultaneously explore and exploit, as well as its successful application to real-world problems, lead to the conclusion that GA is a strong and flexible optimization tool.

We discussed in this review a source of new research in GAs and the information on each component of GA, with our hope to inspire researchers to learn the foundations of GA and to apply what they have learned to their research issues.

Refrences

[1] . A. E. Eiben and J. E. Smith, Introduction to evolutionary computing, vol. 53, Springer, 2003.

[2] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in 2016 International conference on global trends in signal processing, information computing and communication (ICGTSPICC), 2016.

[3] . P. T. H. Hanh, . P. D. Thanh and H. T. T. Binh, "Evolutionary algorithm and multifactorial evolutionary algorithm on clustered shortest-path tree problem," Information Sciences, vol. 553, pp. 280-304, 2021.

[4] . S. O. Ogundoyin and . I. A. Kamil, "Optimization techniques and applications in fog computing: An exhaustive survey," Swarm and Evolutionary Computation, vol. 66, p. 100937, 2021.

[5] . I. R. Management Association, Nature-Inspired Computing: Concepts, Methodologies, Tools, and Applications, IGI Global, 2016, p. 1780.

[6] T. Back, U. Hammel and Schwefel, H-P, "Evolutionary computation: Comments on the history and current state," IEEE transactions on Evolutionary Computation, vol. 1, no. 1, pp. 3-17, 1997.

[7] A. Slowik and . H. Kwasnicka, "Evolutionary algorithms and their applications to engineering problems," Neural Computing and Applications, pp. 1-17, 2020.

[8] . D. Simoncini and K. Y. Zhang, "Population-Based Sampling and Fragment-Based De Novo Protein Structure Prediction," 2019.

[9] . B. Galvan, D. Greiner, . J. Périaux, . M. Sefrioui and G. Winter, "Parallel Evolutionary Computation for Solving Complex CFD Optimization Problems : A Review and Some Nozzle Applications," Parallel Computational Fluid Dynamics 2002, pp. 573-604, 2003.

[10] . D. Greiner, . J. Periaux, . D. Quagliarella, J. Magalhaes-Mendes and B. Galván, "Evolutionary algorithms and metaheuristics: applications in engineering design and optimization," Mathematical Problems in Engineering, vol. 2018, 2018.

[11] X. Xia, H. Qiu, X. Xu and Y. Zhang, "Multi-objective workflow scheduling based on genetic algorithm in cloud environment," Information Sciences, vol. 606, pp. 38--59, 2022.

[12] P. Hartmut, "Evolutionary Algorithms: Overview, Methods and Operators," GEATbx version, vol. 3, 2006.

[13] . K. Gao, Z. Cao, . L. Zhang, Z. Chen, Y. Han and . Q. Pan, "A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems," IEEE/CAA Journal of Automatica Sinica, vol. 6, no. 4, pp. 904-916, 2019.

[14] A. Nayyar, S. Garg, . D. Gupta and A. Khanna, "Evolutionary computation: theory and algorithms," in Advances in swarm intelligence for optimizing problems in computer science, Chapman and Hall/CRC, 2018, pp. 1-26.

[15] . R. Cheng, C. He, . Y. Jin and X. Yao, "Model-based evolutionary algorithms: a short survey," Complex \& Intelligent Systems, vol. 4, no. 4, pp. 283-292, 2018.

[16] . J. H. Holland and others, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, Cambridge: MIT press, 1992.

[17] D. FLOREANO and J. URZELAI, Evolutionstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution Evolutionstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution, 1973, Frommann-Holzboog, 1973.

[18] . X. Yao, Y. Liu and . G. Lin, "Evolutionary programming made faster," IEEE Transactions on Evolutionary computation, vol. 3, no. 2, pp. 82-102, 1999.

[19] . J. R. Koza, Genetic programming: on the programming of computers by means of natural selection, vol. 1, MIT press, 1992.

[20] R. Storn and . K. Price, "Differential evolution--a simple and efficient heuristic for global optimization over continuous spaces," Journal of global optimization, vol. 11, no. 4, pp. 341-359, 1997.

[21] . M. A. Mohammed, M. Ghani, O. I. Obaid, . S. A. Mostafa , . M. S. Ahmad, . D. A. Ibrahim and . M. Burhanuddin, "A review of genetic algorithm application in examination timetabling problem," Journal of Engineering and Applied Sciences, vol. 12, no. 20, pp. 5166-5181, 2017.

[22] J. H. Holland, Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence, Cambridge, MA: MIT Press, 1975.

[23] M. G. Altarabichi, S. Nowaczyk, S. Pashami and P. Sheikholharam Mashhadi, "Fast Genetic Algorithm for feature selection—A qualitative approximation approach," in Proceedings of the Companion Conference on Genetic and Evolutionary Computation, 2023.

[24] . S. Katoch, S. S. Chauhan and V. Kumar, "A review on genetic algorithm: past, present, and future," Multimedia Tools and Applications, pp. 1--36, 2020.

[25] A. E. Eiben and J. E. Smith, Introduction to evolutionary computing, Springer Science & Business Media, 2003.

[26] D. Whitley, "A genetic algorithm tutorial," Statistics and computing, vol. 4, no. 2, pp. 65-85, 1994.

[27] Y. Liu, D. Cetenovic, H. Li, E. Gryazina and V. Terzija, "An optimized multi-objective reactive power dispatch strategy based on improved genetic algorithm for wind power integrated systems," International Journal of Electrical Power \& Energy Systems, vol. 136, p. 107764.

[28] Y. Qin, Z. Li, J. Ding, F. Zhao and M. Meng, "Automatic optimization model of transmission line based on GIS and genetic algorithm," Array, vol. 17, p. 100266, 2023.

[29] P. W. Tsang and A. T. Au, "A genetic algorithm for projective invariant object recognition," in TENCON'96. Proceedings., 1996 IEEE TENCON. Digital Signal Processing Applications, 1996.

[30] D. E. Golberg, "Genetic algorithms in search, optimization, and machine learning," Addion wesley, 1989.

[31] K. P. Cheng, . R. E. Mohan, . N. H. K. Nhan and . A. V. Le, "Multi-objective genetic algorithm-based autonomous path planning for hinged-tetro reconfigurable tiling robot}," IEEE Access, vol. 8, pp. 121267-121284, 2020.

[32] A. Sharma, . R. Patani and A. Aggarwal, "Software testing using genetic algorithms," International Journal of Computer Science \& Engineering Survey, vol. 7, no. 2, pp. 21--33, 2016.

[33] . Y. Zhang, . P. Li and X. Wang, "Intrusion detection for IoT based on improved genetic algorithm and deep belief network," IEEE Access, vol. 7, pp. 31711--31722, 2019.

[34] T. Barman and N. Deb, "State of the art review of speech recognition using genetic algorithm," in 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), 2017.

[35] . Z. B. Vaishnav and . P. S. Sajja, "Knowledge-based approach for word sense disambiguation using genetic algorithm for Gujarati," in Information and communication technology for intelligent systems, Springer, 2019, pp. 485--494.

[36] . W. Zang, . Z. Wang, D. Jiang, X. Liu and Z. Jiang, "Classification of MRI brain images using DNA genetic algorithms optimized Tsallis entropy and support vector machine," Entropy, vol. 20, no. 12, p. 964, 2018.

[37] C. Hamzaçebi, "Improving genetic algorithms' performance by local search for continuous function optimization," Applied Mathematics and Computation, vol. 196, no. 1, pp. 309--317, 2008.

[38] J. Zhou , S. Huang, T. Zhou, D. J. Armaghani and Y. Qiu, "Employing a genetic algorithm and grey wolf optimizer for optimizing RF models to evaluate soil liquefaction potential," Artificial Intelligence Review, vol. 55, no. 7, pp. 5673--5705, 2022.

[39] K. Sastry, D. Goldberg and . G. Kendall, "Genetic algorithms," in Search methodologies, Springer, 2005, p. Search methodologies.

[40] Z. Michalewicz, Genetic algorithms+ data structures= evolution programs, Springer Science & Business Media, 2013.

[41] . Y. Chen, M. Elliot and J. Sakshaug, "Genetic algorithms in matrix representation and its application in synthetic data," in UNECE Worksession on Statistical Confidentiality 2017, 2017.

[42] B. Morgan, "Towards Data Science," 26 6 2021. [Online]. Available: https://towardsdatascience.com/unit-3-genetic-algorithms-part-1-986e3b4666d7. [Accessed 17 12 2021].

[43] M. Safe, J. Carballido, I. Ponzoni and N. Brignole, "On stopping criteria for genetic algorithms," In Advances in Artificial Intelligence–SBIA, pp. 405-413, 2004.

[44] U. Bodenhofer, "Genetic algorithms: theory and applications," Lecture notes, Fuzzy Logic Laboratorium Linz-Hagenberg, Winter, vol. 2004, 2003.

[45] K.-F. Man, . K.-S. Tang and S. Kwong, "Genetic algorithms: concepts and applications," IEEE transactions on Industrial Electronics, vol. 43, no. 5, pp. 519--534, 1996.

[46] . E. Vonk, L. C. Jain and . R. Johnson, "Using genetic algorithms with grammar encoding to generate neural networks," in Proceedings of ICNN'95-International Conference on Neural Networks, 1995.

[47] A. Hussain, . Y. S. Muhammad, M. Nauman Sajid, I. Hussain, . A. Mohamd Shoukry and . S. Gani, "Genetic algorithm for traveling salesman problem with modified cycle crossover operator," Computational intelligence and neuroscience, vol. 2017, 2017.

[48] A. Singh and L. Jain, "VLSI floorplanning using entropy based intelligent genetic algorithm}," in International Conference on Computing, Analytics and Networks, 2017.

[49] V. Paul, G. C. Victer and . L. Jayakumar, "Performance evaluation of population seeding techniques of permutation-coded GA traveling salesman problems based assessment: performance evaluation of population seeding techniques of permutation-coded GA," International Journal of Applied Metaheuristic Computing (IJAMC), vol. 10, no. 2, pp. 55--92, 2019.

[50] . V. Toğan and T. D. Ayşe , "An improved genetic algorithm with initial population strategy and self-adaptive member grouping," Computers \& Structures, vol. 86, no. 11-12, pp. 1204--1218, 2008.

[51] . E. Alkafaween, . A. B. Hassanat and S. Tarawneh, "Improving Initial Population for Genetic Algorithm using the Multi Linear Regression Based Technique (MLRBT)," Communications-Scientific letters of the University of Zilina, vol. 23, no. 1, pp. E1--E10, 1 2021.

[52] W. Pan, K. Li, M. Wang, . J. Wang and . B. Jiang, "Adaptive randomness: a new population initialization method," Mathematical Problems in Engineering, vol. 2014, 2014.

[53] . A. B. Hassanat, V. Prasath, M. A. Abbadi, . S. A. Abu-Qdari and H. Faris, "An improved genetic algorithm with a new initialization mechanism based on regression techniques," Information, vol. 9, no. 7, p. 167, 2018.

[54] Y. Wei, Y. nd Hu and K. Gu, "Parallel search strategies for TSPs using a greedy genetic algorithm," in Third international conference on natural computation (ICNC 2007), 2007.

[55] . S. S. Ray, S. Bandyopadhyay and S. K. Pal, "Genetic operators for combinatorial optimization in TSP and microarray gene ordering," Applied intelligence, vol. 26, no. 3, pp. 183--195, 2007.

[56] P. V. Paul, . P. Dhavachelvan and . R. Baskaran, "A novel population initialization technique for genetic algorithm," in 2013 International Conference on Circuits, Power and Computing Technologies (ICCPCT), 2013.

[57] . K. Jebari and . M. Madiafi, "Selection methods for genetic algorithms," International Journal of Emerging Sciences, vol. 3, no. 4, pp. 333--344, 2013.

[58] . T. Bäck and F. Hoffmeister, "Extended selection mechanisms in genetic algorithms," 1991.

[59] A. Shukla, H. M. Pandey and D. Mehrotra, "Comparative Review of Selection Techniques in Genetic Algorithm," 2015.

[60] D. E. Goldberg and . K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in Foundations of genetic algorithms, vol. 1, Elsevier, 1991, pp. 69--93.

[61] K. Jebari, . M. Madiafi and . A. Elmoujahi, "Parent Selection Operators for Genetic Algorithms," International Journal of Engineering, vol. 2, 2013.

[62] . E.-u. Haq , . I. Ahmad, . A. Hussain and I. M. Almanjahie, "A novel selection approach for genetic algorithms for global optimization of multimodal continuous functions," Computational intelligence and neuroscience, vol. 2019, 2019.

[63] D. Pedersen, "Coarse-grained parallel genetic algorithms: Three implementations and their analysis," 1998.

[64] . D. E. Goldberg and . K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in Foundations of genetic algorithms, vol. 1, Elsevier, 1991, pp. 69--93.

[65] M. R. Noraini and J. Geraghty, "Genetic algorithm performance with different selection strategies in solving TSP," in Proceedings of the World Congress on Engineering 2011 Vol II, London, U.K, 2011.

[66] R. Champlin, "Selection Methods of Genetic Algorithms," 2018.

[67] . C.-Y. Lee, "Entropy-Boltzmann selection in the genetic algorithms," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 33, no. 1, pp. 138--149.

[68] P. Sharma and A. Wadhwa, "Analysis of Selection Schemes for Solving an Optimization Problem in Genetic Algorithm," International Journal of Computer Applications, vol. 93, no. 11, 2014.

[69] . A. B. Hassanat and . E. Alkafaween, "On enhancing genetic algorithms using new crossovers," International Journal of Computer Applications in Technology, vol. 55, no. 3, pp. 202--212, 2017.

[70] E. O. Alkafaween, "Novel methods for enhancing the performance of genetic algorithms," Mutah university, jordan, 2018.

[71] K. F. Man, K. S. Tang and S. Kwong, "Genetic Algorithms: Concepts and Applications," IEEE Transactions on Industrial Electronics, vol. 43, no. 5, pp. 519-534, 1996.

[72] Y. Kaya and M. Uyar, "A novel crossover operator for genetic algorithms: ring crossover," arXiv preprint arXiv:1105.0355, 2011.

[73] G. Syswerda, "Uniform crossover in genetic algorithms," in Genetic Algorithms, 1989.

[74] J. Grefenstette, R. Gopal, B. Rosmaita and D. Van Gucht, "Genetic algorithms for the traveling salesman problem," in Proceedings of the first International Conference on Genetic Algorithms and their Applications (pp. 160-168). Lawrence Erlbaum, New Jersey, 1985.

[75] W. M. Spears and K. A. De Jong, "An analysis of multi-point crossover," Foundations of genetic algorithms, 1990.

[76] I. M. Oliver, D. Smith and J. R. Holland, "Study of permutation crossover operators on the traveling salesman problem," in Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA. Hillsdale, NJ: L. Erlhaum Associates, 1987.

[77] D. E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," in Proceedings of the first international conference on genetic algorithms and their applications , 1985.

[78] . Y. Xue, . Y. Wang, J. Liang and A. Slowik, "A self-adaptive mutation neural architecture search algorithm based on blocks," IEEE Computational Intelligence Magazine, vol. 16, no. 3, pp. 67--78, 2021.

[79] K. Deb and D. Deb, "Analysing mutation schemes for real-parameter genetic algorithms," International Journal of Artificial Intelligence and Soft Computing, vol. 4, no. 1, pp. 1-28, 2014.

[80] S. Wagner, M. Affenzeller, A. Beham, G. K. Kronberger and S. M. Winkler, "Mutation effects in genetic algorithms with offspring selection applied to combinatorial optimization problems," in In Proceeding of 22nd European modeling and simulation symposium EMSS, 2010.

[81] I. Korejo, S. Yang, K. Brohi and Z. U. Khuhro, "Multi-Population Methods with Adaptive Mutation for Multi-Modal Optimization Problems," International Journal on Soft Computing, Artificial Intelligence and Applications (IJSCAI), vol. 2, no. 2, 2013.

[82] W. Banzhaf, "The “molecular” traveling salesman," Biological Cybernetics, vol. 64, no. 1, pp. 7-14, 1990.

[83] N. Soni and T. Kumar, "Study of Various Mutation Operators in Genetic," (IJCSIT) International Journal of Computer Science and Information Technologies, vol. 5, no. 3, pp. 4519-4521, 2014.

[84] D. A. Fogel, "A parallel processing approach to a multiple travelling salesman problem using evolutionary programming," in Proceedings of the Fourth annual Symposium on Parallel Processing, 1990.

[85] M. Z. T I, Genetic Algorithms+ data Structures= Evolutionary Programs, Berlin: Springer, 1992.

[86] A. B. Hassanat, E. Alkafaween, . N. A. Al-Nawaiseh, M. A. Abbadi, . M. Alkasassbeh and M. B. Alhasanat, "Enhancing genetic algorithms using multi mutations: Experimental results on the travelling salesman problem," International Journal of Computer Science and Information Security, vol. 14, no. 7, p. 785, 2016.

[87] E. Alkafaween and A. Hassanat, "Improving TSP solutions using GA with a new hybrid mutation based on knowledge and randomness," Communications-Scientific letters of the University of Zilina, vol. 22, no. 3, pp. 128--139, 2020.

[88] A. Chipperfield, . P. and Fleming, . H. Pohlheim and C. Fonseca, "Genetic algorithm toolbox for use with MATLAB," 1994.

[89] . M. Lozano, . F. Herrera and R. C. José , "Replacement strategies to preserve useful diversity in steady-state genetic algorithms," Information sciences, vol. 178, no. 23, pp. 4421--4433, 2008.

[90] . M. Dong and Y. Wu, "Dynamic crossover and mutation genetic algorithm based on expansion sampling," in International Conference on Artificial Intelligence and Computational Intelligence, 2009.

[91] V. A. Cicirello and . S. F. Smith, "Modeling GA performance for control parameter optimization," in GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference, 2000.

[92] . B. Rajakumar and . A. George, "APOGA: An adaptive population pool size based genetic algorithm," AASRI Procedia, vol. 4, 2013.

[93] O. Roeva, S. Fidanova and M. Paprzycki, "Influence of the population size on the genetic algorithm performance in case of cultivation process modelling," in Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on (pp. 371-376), September2013.

[94] . J. Koljonen and J. T. Alander, "Effects of population size and relative elitism on optimization speed and reliability of genetic algorithms," in Proceedings of the ninth Scandinavian conference on artificial intelligence (SCAI 2006), 2006.

[95] Z. Ma and A. W. Krings, "Dynamic populations in genetic algorithms," in Proceedings of the 2008 ACM symposium on Applied computing, 2008.

[96] S. G. B. Rylander and B. Gotshall, Population, p. 3.

[97] . S. G. B. Rylander and B. Gotshall, "Optimal population size and the genetic algorithm," Population, vol. 100, no. 400, p. 900, 2002.

[98] . J. Arabas, Z. Michalewicz and J. Mulawka, "GAVaPS-a genetic algorithm with varying population size," in Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, 1994.

[99] A. E. Eiben, E. Marchiori and . V. Valko, "Evolutionary algorithms with on-the-fly population size adjustment," in International Conference on Parallel Problem Solving from Nature, 2004.

[100] Raghavendra BV, "Effect of Crossover Probability on Performance of Genetic Algorithm in Scheduling of Parallel Machines for BI- Criteria Objectives," International Journal of Engineering and Advanced Technology, vol. 9, no. 1, pp. 2827-2831, 2019.

[101] W. Xiao, . L. Wu, X. Tian and J. Wang, "Applying a new adaptive genetic algorithm to study the layout of drilling equipment in semisubmersible drilling platforms," Mathematical Problems in Engineering, vol. 2015, 2015.

[102] . A. Hassanat, . K. Almohammadi, E. Alkafaween, . E. Abunawas, . A. Hammouri and V. Prasath, "Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach," Information, vol. 10, no. 12, p. 390, 2019.

[103] . M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," IEEE Transactions on Systems, Man, and Cybernetics, vol. 24, no. 4, pp. 656--667, 1994.

[104] W.-Y. Lin, W.-Y. Lee and T.-P. Hong, "Adapting crossover and mutation rates in genetic algorithms," J. Inf. Sci. Eng., vol. 19, no. 5, pp. 889--903, 2003.

[105] . A. Laoufi, S. Hadjeri and A. Hazzab, "Adaptive probabilities of crossover and mutation in genetic algorithms for power economic dispatch," International Journal of Applied Engineering Research, vol. 1, no. 3, pp. 393--408, 2006.

[106] S. Z. Ramadan, "Reducing premature convergence problem in genetic algorithm: Application on travel salesman problem," Computer and Information Science, vol. 6, no. 1, p. 47, 2013.

[107] . S. Malik and S. Wadhwa, "Preventing premature convergence in genetic algorithm using DGCA and elitist technique," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, no. 6, 2014.

[108] . T. N. Fatyanosa and M. Aritsugi, "An Automatic Convolutional Neural Network Optimization Using a Diversity-Guided Genetic Algorithm," IEEE Access, vol. 9, pp. 91410--91426, 2021.

[109] . A. Aldallal, "Avoiding Premature Convergence of Genetic Algorithm in Informational Retrieval Systems," International Journal of Intelligent Systems and Applications in Engineering, vol. 2, no. 4, pp. 80--85, 2014.

[110] . H. M. Pandey, "Secure medical data transmission using a fusion of bit mask oriented genetic algorithm, encryption and steganography," Future Generation Computer Systems, vol. 111, pp. 213--225, 2020.

[111] . J. Xin, . J. Zhong, . F. Yang, Y. Cui and J. Sheng, "An improved genetic algorithm for path-planning of unmanned surface vehicle," Sensors, vol. 19, no. 11, p. Multidisciplinary Digital Publishing Institute, 2019.

[112] . K. Peng, J. Du, . F. Lu, Q. Sun, Y. Dong, . P. Zhou and M. Hu, "A hybrid genetic algorithm on routing and scheduling for vehicle-assisted multi-drone parcel delivery," IEEE Access, vol. 7, pp. 49191--49200, 2019.

[113] Y. Su, . X. Chu, D. Chen and X. Sun, "A genetic algorithm for operation sequencing in CAPP using edge selection based encoding strategy," Journal of Intelligent Manufacturing, vol. 29, no. 2, pp. 313--332, 2018.

[114] E. S. Nicoară, "Mechanisms to avoid the premature convergence of genetic algorithms," Petroleum–Gas University of Ploieşti Bulletin, Math.–Info.–Phys, pp. 87-96, 2009.

[115] . D.-C. Dang, . T. Friedrich, T. Kötzing, M. S. Krejca, P. Kristian Lehre, P. S. Oliveto, D. Sudholt and A. M. Sutton, "Escaping local optima using crossover with emergent diversity," IEEE Transactions on Evolutionary Computation, vol. 22, no. 3, pp. 484--497, 2017.

[116] . M. L. Mauldin, "Maintaining Diversity in Genetic Search," in AAAI, 1984.

[117] P. Bosman and . A. P. Engelbrecht, "Diversity rate of change measurement for particle swarm optimisers," in International Conference on Swarm Intelligence, 2014.

[118] J. Mwaura, . A. P. Engelbrecht and F. V. Nepomuceno, "Diversity Measures for Niching Algorithms," Algorithms, vol. 14, no. 2, p. 36, 2021.

[119] J. VasconcelosX, . J. A. Ramirez, R. Takahashi and R. Saldanha, "Improvements in genetic algorithms," IEEE Transactions on magnetics, vol. 37, no. 5, pp. 3414--3417, 2001.

[120] L. C. Hern'andez, A. M. Hern'andez, G. M. C. Cardoso and . Y. M. Jim'enez, "Genetic algorithms with diversity measures to build classifier systems," Investigacion Operacional, vol. 36, no. 3, pp. 206--224, 2015.

[121] D. Gupta and S. Ghafir, "An overview of methods maintaining diversity in genetic algorithms," International journal of emerging technology and advanced engineering, vol. 2, no. 5, pp. 56-60, 2012.

[122] P. Siarry, A. Pétrowski and M. Bessaou, "A multipopulation genetic algorithm aimed at multimodal optimization," Advances in Engineering Software, vol. 33, no. 4, pp. 207-213, 2002.

[123] D. Whitley, S. Rana and R. B. Heckendorn, "The island model genetic algorithm: On separability, population size and convergence," Journal of Computing and Information Technology, vol. 7, pp. 33-48, 1999.

[124] S. Yang, "PDGA: the primal-dual genetic algorithm," Design and Application of Hybrid Intelligent Systems, pp. 214-223, 2003.

[125] T. Park and K. R. Ryu, "A dual-population genetic algorithm for balanced exploration and exploitation," Computational Intelligence, pp. 90-95, 2006.

[126] L. Booker, "Improving search in genetic algorithms," Genetic algorithms and simulated annealing, pp. 61-73, 1987.

[127] . B. Sareni, . L. Krahenbuhl and A. Nicolas, "Niching genetic algorithms for optimization in electromagnetics. I. Fundamentals," IEEE transactions on magnetics, vol. 34, no. 5, pp. 2984--2987, 1998.

[128] T. Bäck and H. P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," Evolutionary computation, vol. 1, no. 1, pp. 1-23, 1993.

[129] M. Nowostawski and R. Poli, "Parallel genetic algorithm taxonomy," in Knowledge-Based Intelligent Information Engineering Systems, 1999. Third International Conference, 1999.

[130] W. E. Hart, E. Baden, R. K. Belew and S. Kohn, "Analysis of the numerical effects of parallelism on a parallel genetic algorithm," in Parallel Processing Symposium, 1996., Proceedings of IPPS'96, The 10th International, 1996.

[131] E. Cantú-Paz, "A survey of parallel genetic algorithms," Calculateurs paralleles, reseaux et systems repartis, vol. 10, no. 2, pp. 141-171, 1998.

[132] . F. De Toro, J. Ortega, J. Fern'andez and A. Díaz, "PSFGA: a parallel genetic algorithm for multiobjective optimization," in Proceedings 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, 2002.

[133] M. Golub and L. Budin, "An asynchronous model of global parallel genetic algorithms," in Second ICSC Symposium on Engineering of Intelligent Systems EIS2000. Hrvatska znanstvena bibliografija i MZOS-Svibor, 2000.

[134] J. Sarma and K. De Jong, "An analysis of the effects of neighborhood size and shape on local selection algorithms," in Parallel Problem Solving From Nature—PPSN IV, 1996.

[135] S. Baluja, "Structure and Performance of Fine-Grain Parallelism in Genetic Search," ICGA, pp. 155-162, 1993.

[136] . T. A. El-Mihoub, A. A. Hopgood, L. Nolle and . A. Battersby, "Hybrid Genetic Algorithms: A Review.," Eng. Lett., vol. 13, no. 2, pp. 124--137, 2006.

[137] W. Wan and . J. B. Birch, "An improved hybrid genetic algorithm with a new local search procedure," Journal of Applied Mathematics, vol. 2013, 2013.

[138] P. Preux and . E.-G. Talbi, "Towards hybrid evolutionary algorithms," International transactions in operational research, vol. 6, no. 6, pp. 557--570, 1999.

[139] H. Dong, T. Li, . R. Ding and J. Sun, "A novel hybrid genetic algorithm with granular information for feature selection and optimization," Applied Soft Computing, vol. 65, pp. 33--46, 2018.

[140] . R. K. Arakaki and . F. L. Usberti, "Hybrid genetic algorithm for the open capacitated arc routing problem," Computers \& Operations Research, vol. 90, pp. 221--231, 2018.

[141] . G. T. Reddy, M. P. K. Reddy, K. Lakshmanna, D. S. Rajput, R. Kaluri and G. Srivastava, "Hybrid genetic algorithm and a fuzzy logic classifier for heart disease diagnosis," Evolutionary Intelligence, vol. 13, no. 2, pp. 185--196, 2020.

[142] A. Oztekin, . L. Al-Ebbini, Z. Sevkli and D. Delen, "A decision analytic approach to predicting quality of life for lung transplant recipients: A hybrid genetic algorithms-based methodology," European Journal of Operational Research, vol. 266, no. 2, pp. 639--651, 2018.

[143] T. Murata and . H. Ishibuchi, "MOGA: multi-objective genetic algorithms," in IEEE international conference on evolutionary computation, 1995.

[144] . C. Fonseca and . P. Fleming, "Multiobjective genetic algorithms," in IEE colloquium on genetic algorithms for control systems engineering, 1993.

[145] N. Spolaor, A. C. Lorena and H. Diana Lee, "Feature selection via pareto multi-objective genetic algorithms," Applied Artificial Intelligence, vol. 31, no. 9-10, pp. 764--791, 2017.

[146] . E. A. K. Mishra, E. Y. Mohapatra and . E. A. K. Mishra, "Multi-Objective Genetic Algorithm: A Comprehensive Survey," International Journal of Emerging Technology and Advanced Engineering, vol. 3, no. 2, pp. 81--90, 2013.

[147] . N. A. Zolpakar, . S. S. Lodhi, . S. Pathak and . M. A. Sharma, "Application of multi-objective genetic algorithm (MOGA) optimization in machining processes," in Optimization of Manufacturing Processes, Springer, 2020, pp. 185--199.

[148] K. Amouzgar, "Multi-objective optimization using genetic algorithms," 2012.

[149] L. d. P. A. Sales, A. R. Pitombeira-Neto and B. de Athayde Prata, "A genetic algorithm integrated with Monte Carlo simulation for the field layout design problem}," Oil & Gas Sciences and Technology--Revue d'IFP Energies nouvelles, vol. 73, p. 24, 2018.

[150] N. Casas, "Genetic algorithms for multimodal optimization: a review," arXiv preprint arXiv:1508.05342, 2015.

[151] M. Stelmack, N. Nakashima and S. Batill, "Genetic algorithms for mixed discrete/continuous optimization in multidisciplinary design," in 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 1998.

[152] B. H. Arabi, "Solving np-complete problems using genetic algorithms," in 2016 UKSim-AMSS 18th International Conference on Computer Modelling and Simulation (UKSim), 2016.

[153] N. Shor, N. Zhurbenko, A. Likhovid and P. Stetsyuk, "Algorithms of nondifferentiable optimization: Development and application," Cybernetics and Systems Analysis, vol. 39, pp. 537--548, 2003.

[154] T. Harada and E. Alba, "Parallel genetic algorithms: a useful survey," ACM Computing Surveys (CSUR), vol. 53, no. 4, pp. 1--39, 2020.

[155] A. Konak, D. W. Coit and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," Reliability engineering \& system safety, vol. 91, no. 9, pp. 992--1007, 2006.

[156] D. B. Fogel, "An introduction to simulated evolutionary optimization," IEEE transactions on neural networks, vol. 5, no. 1, pp. 3--14, 1994.

[157] J. A. Lima, N. Gracias, H. Pereira and A. Rosa, "Fitness function design for genetic algorithms in cost evaluation based problems," in Proceedings of IEEE International Conference on Evolutionary Computation, 1996.

[158] A. Vié, "Qualities, Challenges and Future of Genetic Algorithms," Available at SSRN 3726035, 2020.

[159] S. Droste and D. Wiesmann, On representation and genetic operators in evolutionary algorithms, Citeseer, 1998.

[160] R. L. Haupt, "Optimum population size and mutation rate for a simple real genetic algorithm that optimizes array factors," in IEEE Antennas and Propagation Society International Symposium. Transmitting Waves of Progress to the Next Millennium. 2000 Digest. Held in conjunction with: USNC/URSI National Radio Science Meeting (C}, 2000.