Evolutionary Intelligence
https://doi.org/10.1007/s12065-023-00822-6

REVIEW ARTICLE

Genetic algorithms: theory, genetic operators, solutions, and applications

Bushra Alhijawi[1] · **Arafat Awajan**[1,2]

Received: 24 July 2022 / Revised: 2 December 2022 / Accepted: 31 December 2022

Abstract
A genetic algorithm (GA) is an evolutionary algorithm inspired by the natural selection and biological processes of reproduction of the fittest individual. GA is one of the most popular optimization algorithms that is currently employed in a wide range of real applications. Initially, the GA fills the population with random candidate solutions and develops the optimal solution from one generation to the next. The GA applies a set of genetic operators during the search process: selection, crossover, and mutation. This article aims to review and summarize the recent contributions to the GA research field. In addition, the definitions of the GA essential concepts are reviewed. Furthermore, the article surveys the real-life applications and roles of GA. Finally, future directions are provided to develop the field.

Keywords Genetic algorithms · GA · Evolutionary algorithm · Genetic operators · Optimization

1 Introduction

An optimization is a mathematical tool that selects the best solution from available alternatives. Several real-world mathematically defined problems use the optimization concept, such as scheduling, engineering, mathematics, commerce, networks, and economics. Within the last decades, solving optimization problems has caught researchers' attention. Metaheuristic optimization algorithms (MOA) are commonly utilized to solve those problems [1]. MOA can be classified based on a search strategy (local search and global search), the number of candidate solutions (single solution and population-based), and hybridization (hybrid and memetic). There are several kinds of MOA, such as Evolutionary Algorithms (EAs) and Swarm intelligence [2–5]. Formally, the MOA is given in the following form.

Algorithm 1 General Procedure of MOA.
Let X be a set of possible solutions.
Find $x_0 \in X$ such that maximize/minimize f
(i.e. $f(x_0) = max_{x \in X}(f(x))$ or $f(x_0) = min_{x \in X}(f(x))$)
where $f : X \rightarrow R$.

EAs are the most well-known global-search, population-based, and memetic MOAs. EAs are heuristic methods inspired by mechanisms that rely on biological evolution, such as reproduction, mutation, and natural selection. In EA, the search space X is a set of chromosomes (i.e., DNA strings) considered candidate solutions for a specific problem. Their fitness is evaluated by objective function (f). The fitness value of $x \in X$ should be kept as high (or low) as

✉ Bushra Alhijawi
b.alhijawi@psut.edu.jo

Arafat Awajan
awajan@psut.edu.jo

[1] Princess Sumaya University for Technology, Amman, Jordan

[2] Mutah University, Al-Karak, Jordan

Published online: 03 February 2023

possible to find the optimal solution (i.e., the fittest chromosome). Numerous EAs are developed, such as Genetic Algorithms (GAs) [6–8], Genetic Programming (GP) [9], Evolutionary Programming (EP) [10], Deferential Evolution [11] and Evolution Strategy (ES) [12].

John Holland, his students, and colleagues developed the GAs in the 1960s and 1970s at the University of Michigan [13]. The GAs mimic natural selection (i.e., survival of the fittest) and the biological reproduction processes of the fittest individual. The optimal solution (i.e., the fittest individual) is developed from one generation to the next without depending on strict mathematical formulation [14]. Therefore, the optimal solution consists of the best components (i.e., genes) of the fittest individual in previous generations. Therefore, a GA is considered a nonlinear, discrete event and stochastic process rather than a mathematically guided algorithm [14].

The simplest form of GA works on a population consisting of individuals (i.e., fixed bit stings). GA selects a parent pool from the population based on selection criteria to prepare the next generation. The crossover and mutation operators supply the population with new candidates. The crossover operator produces new children (i.e., offspring) by exchanging partial bit strings and inverting bits between two distinct parents. The mutation operator may flip some genes of the new children. GA evaluates each individual using a fitness function. In the last generation, the fittest individual is considered the optimal solution.

This systematic survey reviews the literature on advances in GAs and their applications. It provides a panorama through which readers can quickly understand the current state of GA research progress and applications. It directly serves researchers and professionals interested in and involved in this research area. The main objectives of this survey are four-fold: (1) We review the essential components of GA and different techniques used in the literature. (2) We survey the recent contributions to the GA research field. (3) We survey the applications and roles of the GA in real-life. (4) We identify directions for future research in this area.

We collected over 90 articles on GAs published in 2011–2021. Searches are conducted on the ACM Digital Library, Springer Link, Web of Science, and ScienceDirect to identify the relevant literature. We also covered the most popular related conferences to collect recent research. The research articles are searched using a binary combination of major keywords: genetic algorithm, genetic operator, crossover operator, mutation operator, evolutionary algorithm, population initialization, and optimization.

The rest of the survey article is organized as follows. The following section presents the essential terminologies of GAs. Section 3 details the essential components of GA. Section 4 reviews the recent contributions in the GAs research area. Section 5 shows real-world applications of the GA and discusses its roles. Finally, Sect. 6 concludes the articles and provides future directions.

2 Terminologies and definitions

The terminologies of GA are borrowed from biology as it is a simulation of a biological process. However, the GA's entities are much simpler than the real biological terminologies [15]. The fundamental terminologies of GAs are:

- *Population* A population is a set of candidate solutions.
- Chromosome/individual. A chromosome/individual is a candidate solution. Each chromosome consists of a set of *genes* and their *alleles*. A gene is one element position of a chromosome, which is a single bit or short block of adjacent bits [15]. An allele is the gene's value of a particular chromosome. Section 3.1 discusses more details about individual representation.
- *Initialization* Initialization is the first process in GA responsible for preparing the initial population. The GA fills the population with random candidate solutions (i.e., individuals). Section 3.1 illustrates more details about the initialization phase.
- Evaluation. An evaluation process is responsible for determining the fitness level of an individual. GA utilizes a problem-dependent fitness function. This operation is triggered once a new individual is produced. Section 3.2 presents more details about the individual evaluation process.
- *Selection* A selection process is essential in GA to select the parents for the crossover operation. The simplest selection technique is based on the fitness value, where the better solutions have the highest probability of being selected than the worse ones. Section 3.3.1 discusses other selection techniques.
- *Crossover* A crossover operation is a recombination process responsible for producing new offspring. Section 3.3.2 presents the variations of crossover.
- *Mutation* A mutation operation is a random deformation of the individual with a specific probability. Section 3.3.3 shows different variations of mutation.
- *Replacement* A replacement operation is responsible for preparing the population for the next generation. The basic technique selects the fittest individuals of the current generation (i.e., parents and new offspring) to prepare the next generation. Section 3.4 discusses the other replacement techniques.
- *Stop criteria* Stop criteria are specified to determine when to stop the GA and select the optimal solution. Typically, at least one of the following criteria is specified:

- Reach the maximum number of generations.
- Find an individual in the population with a fitness value lower/higher than a threshold.

3 Genetic algorithm basics and operations

A GA starts with initializing a population of size *N*. The fitness value of each individual in the population is evaluated using a fitness function. Then, the process enters a loop for a specified number of generations (*maxGan*) where GA uses the current generation (*currentGen*) to generate the next generation. A set of individuals is selected to form the parents pool. The parent pool helps in producing new offspring using crossover and mutation techniques. The search process is terminated when a stop criterion is satisfied (i.e., reaching the *maxGan* generation or finding an individual who satisfied a stop criterion). Algorithm 2 defines the typical GA.

- Population diversity. A population with a low diversity level leads to a GA like a local search algorithm with an additional overhead from maintaining many similar solutions [16]. *Premature convergence* refers to a population containing similar individuals before exploring the search space. A diverse population helps the GA explore different regions of the search space, thus reducing the probability of being stuck in the local optimum of a bad fitness degree.
- Population size. The population size is fixed; thus, it significantly impacts GA performance. The probability of covering promising regions of the search space decreases as the search space dimensionality increases [17]. Thus, selecting a small population size reduces population diversity quickly after applying the crossover operation. On the other hand, selecting a large population waste computational resources.

Algorithm 2 Typical GA.

```
Generate the initial population (Pop(0))
maxGen ← Maximum number of generation
currentGen ← 1
while currentGen < maxGen do
    Evaluate Pop(currentGen -1)          (apply the fitness function)
    Produce Pop(currentGen) depends
      on Pop(currentGen -1)              (apply the genetic operators)
    currentGen ← currentGen + 1
end while
```

3.1 Initialization and individual representation

The population is a set of candidate solutions (Fig. 1). The GA utilizes the population to investigate various search space areas that facilitate global exploration. Therefore, the quality of the initial population has a critical impact on the performance of GA. Two essential characteristics control the population quality:

$$Popualtion = \begin{bmatrix} ind_1 \\ ind_2 \\ ind_3 \\ . \\ . \\ . \\ ind_n \end{bmatrix} = \begin{bmatrix} G_1^1, G_2^1, G_3^1, ..., G_m^1 \\ G_1^2, G_2^2, G_3^2, ..., G_m^2 \\ G_1^3, G_2^3, G_3^3, ..., G_{size}^3 \\ . \\ . \\ . \\ G_1^n, G_2^n, G_3^n, ..., G_m^n \end{bmatrix}$$

Fig. 1 The GA's population

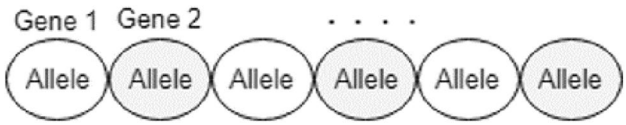

Fig. 2 An individual

1. Randomness criterion. The initialization process produces unpredictable and irregular individuals. Randomness-based initialization techniques are [17]:
   - Stochastic technique generates different populations based on different initial seeds.
   - Deterministic technique generates the same population regardless of the initial seed.
2. Compositionality criterion. Compositionality-based approaches are two types according to the number of standalone procedures involved in the initialization process.
   - Non-compositional technique is the simplest initialization technique that produces populations in a single step.
   - Compositional technique utilizes more than one step to produce the initial population.
3. Generality criterion. Initialization techniques are two types based on generality:
   - Generic technique is not a domain-dependent initialization approach.
   - Specific application technique is a technique initializing the population of a specific problem.

The representation of the individual in the population depends mainly on the problem. An individual represents by using a bit string (i.e., simplest and most popular encoding) or non-binary representation. Generally, the individual consists of a set of genes, and each gene has an allele (Fig. 2). The genes' values (i.e., alleles) are determined initially in the initialization step.

3.2 Fitness function

The fitness function is an essential component of any GA used to measure the fitness value of individuals. A fitness function depends on a single objective function or multi-objective function. The objective function is a function that measures the performance concerning a set of parameters (i.e., alleles of the individual). In contrast, the fitness function measures the reproduction probability of each individual depending on the objective function [18]. Equation 1 defines the fitness function of the simplest form of GA.

$$FF_x = \frac{f_x}{\bar{f}} \tag{1}$$

where

- f_x refers to the objective function value of individual x.
- $\bar{f}$ refers to the average of the objective function values of all individual in the population.

Many problems have one objective that minimizes or maximizes a single objective function. Maximizing accuracy or minimizing the distance are examples of objectives. However, multiple objectives must be optimized simultaneously to solve real-world problems [19]. In such cases, the fitness value computation of an individual becomes more complicated.

3.3 Genetic operators

Genetic operators maintain population diversity. GA applies three genetic operators that should cooperate with one another to develop a high-performance GA: *Selection*, *Crossover*, and *Mutation*.

3.3.1 Selection operator

The selection operator prepares the parents' pool for mating. Thus, the selection operator guides the GA to the optimal solution by preferring the fittest individuals over

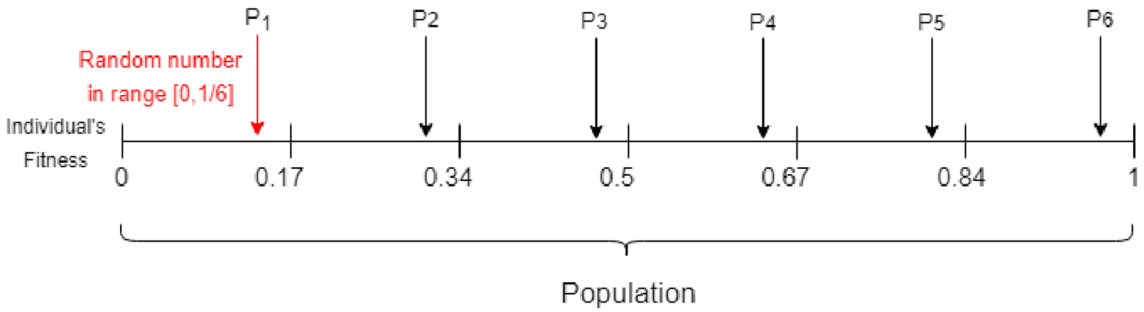

Fig. 3 Stochastic universal selection technique

low-fitted ones [20]. The selection techniques are two types: fitness proportionate selection and ordinal selection [21].

1. Fitness proportionate selection techniques depend on the fitness degree of the individual.
   - Roulette-wheel selection [13, 22]. A selection probability is computed for each individual (Eq. 2). Then, individuals with the highest probability are selected to fill the parent pool.

$$p_i = \frac{f_i}{\sum_{x=1}^{N} fx} \tag{2}$$

   - Stochastic universal selection [23]. The stochastic universal selection uses M equally spaced pointers, where M is the number of parents required to be selected. GA orders the population and selects a single random number ($P_1 \in [0, 1/\mathrm{M}]$). Then, the M parents (i.e., individuals) are chosen starting with P_1 and spaced by $1/M$ (Fig. 3). In roulette-wheel selection, the weaker individuals have a lower probability of being selected, while the stochastic universal selection reduces the unfair nature of the roulette-wheel selection technique.
   - Elitist selection. The fittest M individuals are selected to fill the parents pool.
2. Ordinal Selection.
   - Tournament selection [24]. s individuals are chosen randomly from the population. Then, the fittest selected individual wins the tournament. The common value of s is 2 [21]. Thus, the tournament selection is repeated M times to select M parents.
   - Truncation selection [25]. The individuals are ordered according to fitness value. Then, a particular portion (p) of the fittest individuals are selected and reproduced $1/p$ times.

3.3.2 Crossover operator

A crossover operator is a powerful tool for producing new offspring and improving the quality of individuals by swapping genes between the parents. Several techniques are utilized to perform the crossover.

- One-point crossover is the simplest crossover technique (Fig. 4). One cross point is chosen randomly, and swapped the two tails of the parents to produce two new children.

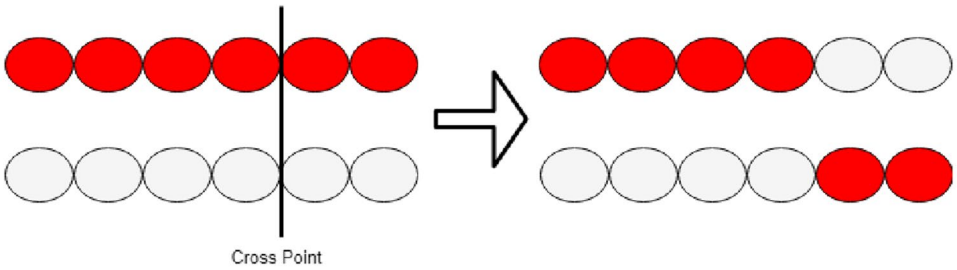

Fig. 4 One-point crossover

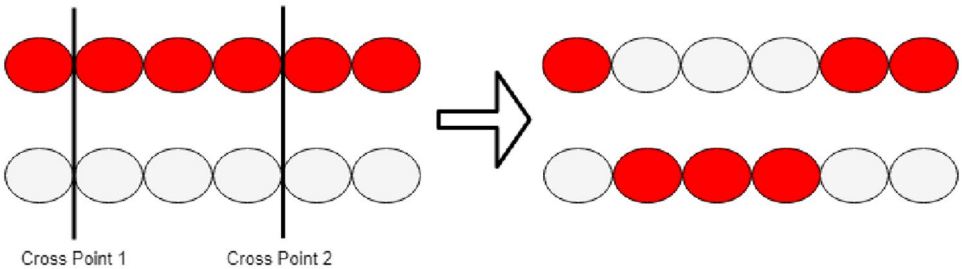

Fig. 5 Two-point crossover

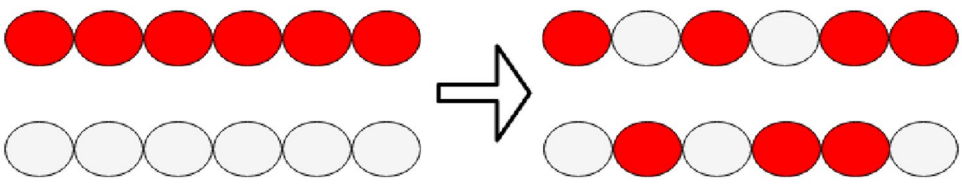

Fig. 6 Uniform crossover

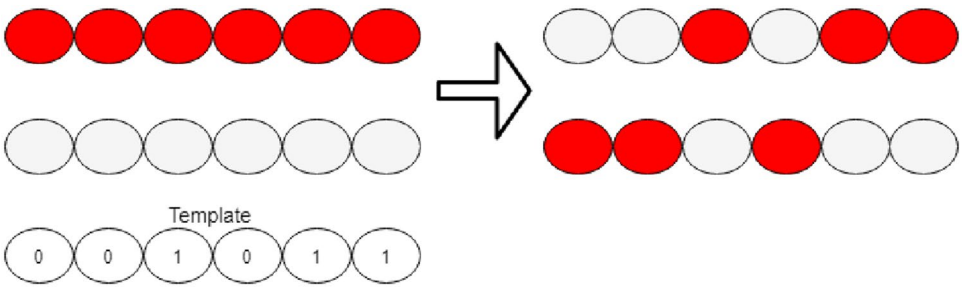

Fig. 7 Uniform order-based crossover

- Two-point crossover (Fig. 5). Two random cross points are selected, and the parents' genes between the two points are exchanged to produce two new children.
- K-point crossover. One-point and two-point crossover techniques are special cases of K-point crossover. Thus, the concept of one-point and two-point crossover is extended to K-point crossover, where $K > 2$ cross points are selected randomly.
- Uniform crossover (Fig. 6) [26] . Uniform crossover uses a fixed mixing ratio (P_c) between two parents. For each gene (g_i) in the child, a random number ($r \in [0, 1]$) is selected. In case $r > P_c$, then gene g_i inherits the allele of the first parent. Otherwise, gene g_i inherits the allele of the second parent.
- Uniform order-based crossover (Fig. 7) [27] . The uniform order-based crossover uses a random binary template to produce children. The first child inherits the genes from the first parent according to the ones in the template; otherwise, it inherits the genes from the second parent. The second child inherits the inverse genes of the first child from both parents.

3.3.3 Mutation operator

Crossover operators negatively impact population diversity since the new children have identical alleles to their parents. The mutation operator maintains the population diversity. The main idea is to change the allele of the child randomly. The mutation operator is controlled by a mutation probability (P_m) that is kept as low as possible to avoid the GA behaving like a random search. Below, the classical mutation techniques are mentioned:

- Single-point mutation selects one random gene and changes it to a random value with a probability P_m.
- Bitwise inversion mutation inverters the whole bit string bit by bit with probability P_m.
- Random variable mutation selects a random number ($K > 1$) of genes and changes it to random values with a probability P_m.
- Boundary mutation is utilized with integer and float alleles. A random gene is selected with probability P_m and randomly changed its value to the lower or upper bound.
- Uniform mutation changes the value of the chosen gene (with probability P_m) to a uniform random value selected between the upper and lower bounds for that gene.

3.4 The replacement techniques

After producing the new offspring, the GA prepares a new population for the next generation. Simply, the GA prepares the next population by selecting the new offspring and the parents who did not undergo crossover to be the next generation. In addition to this method, there are many ways to do this:

- Delete-all (Generational) replacement technique replaces all the current generation (i.e., parents) with the new offspring to prepare the next generation [21]. This method hopes that the children will improve the population fitness on average.
- Steady state replacement technique replaces a small fraction (n) of the population during each iteration with n new individuals.
- Elitism replacement technique selects the fittest individuals among the parents and the new offspring to prepare the next generation.
- Delete n-last replacement technique replaces the worst n individuals of the current population with n individuals from the new offspring group.

4 Recent contributions to genetic algorithms

Several contributions are made to the GAs research area in the last decade. New initialization [28–30], selection [31–34], crossover [35–47] and mutation [42, 48–51] techniques are proposed in that context. Also, new ideas are developed in different aspects, such as the evaluation

process aspect [52] and maintaining the population diversity [38].

Initializing a good population leads to improving the overall performance of the GAs. Therefore, several research efforts focused on the initialization process of the population. Paul et al. [28] proposed a new specific application population seeding technique called Ordered Distance Vector (ODV) for Traveling Salesman Problem (TSP). ODV-based population seeding depends on the ODV matrix (i.e., a matrix contains the distance between the elements of a set). Their technique generates a random and diverse population using Equi-begin with variable diversity, Vari-begin with equal diversity, and Vari-begin with variable diversity methods. Deng et al. [29] developed a population initialization method using the k-mean clustering and a GA. K-mean clustering groups the genes into k clusters. Then, the GA finds the local and global optimum for each group. The initial population is produced depending on the global and local optimum in each group. Hassanat et al. [30] proposed a new method for the initial population seeding based on linear regression analysis for TSP.

Another factor that may improve the overall performance of the GAs is producing high-quality new offspring. Two factors control the quality of new offspring: the selected parents and the crossover technique. Kaya [31] proposed a new selection technique called back controlled selection operator (BCSO). BCSO compares the fitness values of the individual in the current generation with that in the previous generation. Only the individuals with a higher fitness value than that in the previous generation are selected. Rafsanjani and Eskandari [32] integrated roulette wheel, tournament, and BCSO methods to develop Random Combinational Selection Operator (RCSO). RCSO generates a selection probability of each operator to determine the selection operator used in each generation. Rafsanjani and Eskandari [33] proposed a selection operator called sequential combinational selection operator (SCSO). SCSO is similar to RCSO, where SCSO applies the tournament selection technique when the selected random number is even. Otherwise, SCSO applies the roulette wheel selection technique. Hussain et al. [34] proposed a new selection scheme that is the optimal combination of exploration and exploitation.

Several contributions presented new crossover operators. Kaya et al. [35] developed a new crossover technique called Ring Crossover (RC). RC combines the two parents in a ring form and selects a random cutting point to produce two children (Fig. 8). The first child is created in the clockwise direction, while the second is created in the anti-clockwise direction. Semenkin and Semenkina [36] modified the classical uniform crossover to develop an equiprobable uniform crossover. The probability of inheriting a gene from one parent depends on the parent's fitness value in the equiprobable uniform crossover. The parent pool for this technique is prepared using tournament, roulette-wheel, and elitist selection techniques.

Thakur [37] presented a new crossover technique called double Pareto crossover (DPX) depending on double Pareto distribution [53]. Elsayed et al. [38] developed a multi-parents crossover method that ranks the parents from the best ($\vec{\text{Pind}}_1$) to the worst ($\vec{\text{Pind}}_3$) to produce three children $\vec{\text{Ch}}_1$, $\vec{\text{Ch}}_2$ and $\vec{\text{Ch}}_3$ as follows.

$$\vec{Ch}_1 = P\vec{ind}_1 + \beta * (P\vec{ind}_2 - P\vec{ind}_3) \tag{3}$$

$$\vec{Ch}_2 = P\vec{ind}_2 + \beta * (P\vec{ind}_3 - P\vec{ind}_1) \tag{4}$$

$$\vec{Ch}_3 = \vec{n}P\vec{ind}_3 + \beta * (P\vec{ind}_1 - P\vec{ind}_2) \tag{5}$$

where, β is a random number that follows a normal distribution with mean value μ and standard deviation σ.

Osaba et al. [39] developed a new adaptive multi-crossover population algorithm (AMCPA). AMCPA prioritizes the mutation operator and applies the crossover operator only if it benefits the search process. Thus, the crossover probability p_c varies depending on the search performance of the current and recent generations. The p_c initially is set to zero and modified based on the improvement made in the best solution found using Eq. 6. Thus, the p_c is mathematically updated if no improvements are made in the current generation(i.e., the search process is stuck in a local optimum).

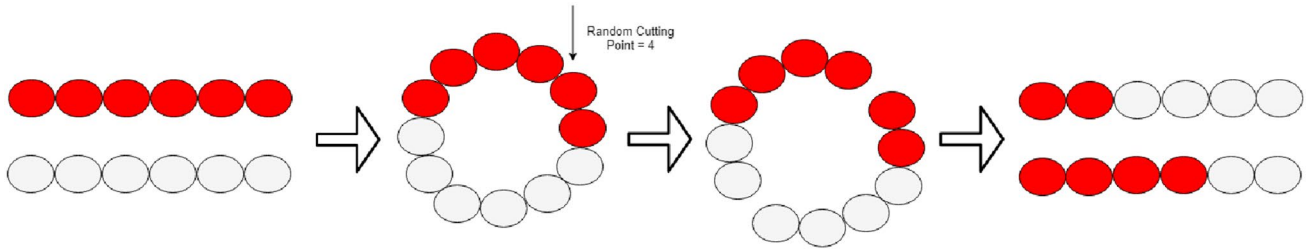

Fig. 8 Ring crossover

$$p_c = p_c + \frac{N^2}{NMF^2} + \frac{NG}{NMF^2} \tag{6}$$

where N^2, NMF^2, and NG refer to the number of generations executed without improvements, the size of the mutation operator neighborhood, and the total number of generations executed, respectively.

An arithmetic crossover is advanced recombination of the parents genes. The genes of new offspring lie between the parents gene values. Jin et al. [40] presented a new arithmetic crossover operator called compound arithmetic crossover (CAC). CAC is utilized in conjunction with a uniform mutation operator. The basic model of CAC is given by:

$$T_s(x_1, x_2) = \begin{cases} a_1 a + (1 - a_1)(x_1 \wedge x_2) & 0 \le r \le r_1 \\ a_2 x_1 + (1 - a_2) x_2 & r_1 \le r \le r_2 \\ a_3 b + (1 - a_3)(x_1 \vee x_2) & r_2 \le r \le 1 \end{cases}$$

where

- x_1 and x_2 are values in the range $[a, b]$ where $x_1 \le x_2$.
- $0 \le r_1 \le r_2 \le 1$ are location parameters.
- a_1, a_2 and $a_3 \in [0, 1]$ and follow the uniform distribution.

Demirc et al. [41] developed CAC based on chaos theory. In Chaos theory, even a small change in the inputs can cause a considerable change in the outputs [41]. The mathematical expression of their crossover is given by:

$$c(x_i) = m * p_1(x_i) + (1 - m) * p_2(x_i) \tag{7}$$

$$m = (0.5 * r) + (0.5 * map) \tag{8}$$

$$map = 4 * z * (1 - z) \tag{9}$$

where

- x_i is a gene (i).
- m is a random number generated by chaotic random number generator (Eq. 8).

Alkafaween [42] presented two new specific application crossover operators for TSP: Cut On Worst Gene Crossover (COWGC) and Cut On Worst L+R Crossover (COWLRGC). COWGC generates new offspring that inherit the parents' best genes to improve fitness value. COWLRGC specifies the worst gene in parents based on the distance between its neighbors (i.e., right and left side genes). Hassanat Alkafaween [43] proposed a new crossover inspired by the principle of the head-on elastic collision called Collision crossover. Collision crossover considers the parents as moving objects toward each other, and the new offspring is produced due to the parents colliding.

The population diversity level is a critical problem that faces the GA. The mutation operator aims to overcome this problem. Alkafaween [42] presented new mutation operators:

- Worst Gene With Random Gene Mutation changes the parent's worst gene with another gene to produce a new offspring.
- Worst Gene With Worst Gene Mutation (WGWWGM) exchanges the two worst genes in the parent.
- Worst Gene With the Worst Around the Nearest Neighbor Mutation (WGWWNNM) swaps the worst gene with the nearest gene. The nearest gene is determined based on the problem.
- Worst Gene Inserted Beside Nearest Neighbor Mutation (WGIBNNM) changes the position of the worst gene to be beside the nearest gene.
- Random Gene Inserted Beside Nearest Neighbor Mutation (RGIBNNM) selects and changes the position of a random gene to be beside the nearest gene.

Albayrak and Allahverdi [48] developed a new mutation operator called Greedy Sub Tour Mutation (GSTM). GSTM uses classical and greedy techniques based on several parameters. Marung et al. [49] designed a new mutation operator

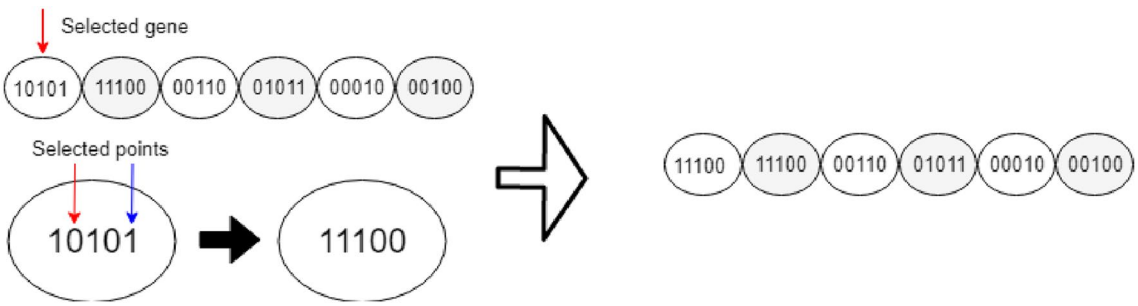

Fig. 9 Mutation operator of [49]

that allows changing positions only within the same gene (i.e., the allele). Two points are selected randomly to interchange their positions of it (Fig. 9).

In order to improve the overall performance of GAs, new ideas are proposed in different aspects than the genetic operators, such as the evaluation process aspect [52] and maintaining population diversity using a diversity operator [38]. Alhijawi [52] proposed a multi-objective GA with an n-fitness function (i.e., n-filtering levels). Each fitness function is related to an objective function where each individual has n-fitness values. The n-filtering levels are ordered based on the domain problem. Each individual should pass one filtering level to reach the other. Thus, the optimal solution should pass n-filtering levels for *maxGan* generations and has the highest fitness value in the last filtering level. Elsayed et al. [38] proposed a new diversity operator instead of the mutation operator. The diversity operator depends on information gathered from the generated offspring using the crossover and from the best m individuals in the current generation.

5 Applications on genetic algorithms

GAs have been used to solve optimization problems in several applications such as image processing [54–57], scheduling [54, 58, 59], clustering [60–62], software engineering [63–66], natural language processing [67, 68], and recommender system (RS) [52, 69, 70].

GAs are used in the fundamental image processing tasks: image enhancement and segmentation. The image quality is enhanced by removing noise, amplifying the level of details, and amplifying image contrast [55]. Many authors used the GAs to enhance image contrast [71, 72]. Hashemi et al. [71] proposed a new technique to enhance image contrast using GA by increasing the visible details and enhancing the contrast of low-illumination images. Daniel and Anitha [72] presented a new optimum green plane masking GA-based technique for contrast enhancement of retinal images. Their GA uses adaptive genetic operators to enhance images dynamically. Deborah and Arymurthy [73] developed a new genetic-based technique to enhance and restore the old document image. Guo et al. [74] used the GA to select the parameter's values of the objective function closely related to a domain problem. The GA aims to construct a customized objective function to obtain the best defogging results for various foggy images.

The GA is applied to find an optimized schedule for several applications. Wijayaningrum and Mahmudy [75] used the GA to assist in optimizing the ship's route scheduling. Xu et al. [76] proposed a GA-based scheduling technique to find the optimal task schedule of heterogeneous computing systems depending on multiple priority queues. Faghihi et al. [77] developed a GA-based scheduling technique to find the best construction project schedule. Konar et al. [78] presented a real-time task scheduling assisted by a GA in a multiprocessor environment. Keshanchia et al. [79] developed a task scheduling in the cloud environments technique using GA.

Also, GAs are used to improve the clustering approaches by finding the optimal clusters' centroids [60, 80, 81] and the optimal clusters (i.e., each chromosome as a set of clusters) [82, 83]. One of the major problems that affect the performance of K-means clustering is selecting the appropriate initial seeds. GAs are utilized to optimize the initial seeds in the K-means algorithm [84–86].

In the software engineering area, GA is used in software testing, software design, and requirements engineering [63]. The main goal of software testing is to design the minimum number of test cases that helps in detecting as many faults as possible [64]. This process is time-consuming and very costly. Therefore, an automated software testing process can significantly reduce cost and time. In this context, the GA generates the test cases [87, 88]. In addition, the GA is utilized in automated cost estimation [65], project planning [65], requirements engineering [63, 65], software design [65, 89] and optimizing source code [65].

RS that uses the GA is called Genetic-based RS. GAs are used in three aspects of RS: clustering [80, 90, 91], hybrid models [92–94], and GA-based CF does not require any additional information [95, 96]. Commonly, the GA finds the optimal similarity measure [96, 97]. GAs have several roles, such as finding the optimal feature weight [98–100] and optimizing the initial seeds in a clustering algorithm [101]. Also, the GA is used to predict the optimal similarity values between users [95]. Alhijawi et al. [52] presented a novel multi-objective genetic-based RS called BLI_GA. BLI_GA employs the GA to find the best list of items for users. Thus, each individual in the population represents a candidate recommendation list.

6 Conclusion

This article reviewed the recent development of GAs and their applications during the last decade. We also surveyed the initialization techniques and genetic operators proposed in the literature. GAs are applied in several applications, such as image processing, scheduling, clustering, software engineering, natural language processing, and RS. There are promising directions for future work in the area. (1) New crossover operators can be developed to improve the quality of individuals. (2) New operators may be developed to improve the diversity of the population. (3) GAs are powerful optimization tools that may be employed in different domains.

References

1. BoussaïD I, Lepagnot J, Siarry P (2013) A survey on optimization metaheuristics. Inf Sci 237:82–117
2. Agushaka JO, Ezugwu AE, Abualigah L (2022) Dwarf mongoose optimization algorithm. Comput Methods Appl Mech Eng 391:114570
3. Ezugwu AE, Agushaka JO, Abualigah L, Mirjalili S, Gandomi AH (2022) Prairie dog optimization algorithm. Neural Comput Appl 34(22):20017–20065
4. Abualigah L, Abd Elaziz M, Sumari P, Geem ZW, Gandomi AH (2022) Reptile search algorithm (RSA): a nature-inspired metaheuristic optimizer. Expert Syst Appl 191:116158
5. Oyelade ON, Ezugwu AE-S, Mohamed TI, Abualigah L (2022) Ebola optimization search algorithm: a new nature-inspired metaheuristic optimization algorithm. IEEE Access 10:16150–16177
6. Holland J, Goldberg D (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Massachusetts
7. Mattfeld DC (2013) Evolutionary search and the job shop: investigations on genetic algorithms for production scheduling
8. Arabali A, Ghofrani M, Etezadi-Amoli M, Fadali MS, Baghzouz Y (2013) Genetic-algorithm-based optimization approach for energy management. IEEE Trans Power Deliv 28(1):162–170
9. Koza JR (1994) Genetic programming as a means for programming computers by natural selection. Stat Comput 4(2):87–112
10. Kim J-H, Myung H (1997) Evolutionary programming techniques for constrained optimization problems. IEEE Trans Evol Comput 1(2):129–140
11. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359
12. Rechenberg I (1973) Evolution strategy: optimization of technical systems by means of biological evolution. Fromman Holzboog Stuttgart 104:15–16
13. Holland JH (1975) Adaptation in natural and artificial systems. In: An introductory analysis with application to biology, control, and artificial intelligence. Ann Arbor: University of Michigan Press, pp 439–444
14. Man K-F, Tang K-S, Kwong S (1996) Genetic algorithms: concepts and applications [in engineering design]. IEEE Trans Ind Electron 43(5):519–534
15. Mitchell M (1998) An introduction to genetic algorithms
16. Sudholt D (2018) The benefits of population diversity in evolutionary algorithms: a survey of rigorous runtime analyses. arXiv preprint arXiv:1801.10087
17. Kazimipour B, Li X, Qin AK (2014) A review of population initialization techniques for evolutionary algorithms. In: 2014 IEEE congress on evolutionary computation (CEC), IEEE, pp 2585–2592
18. Whitley D (1994) A genetic algorithm tutorial. Stat Comput 4(2):65–85
19. Mukhopadhyay A, Maulik U, Bandyopadhyay S, Coello CAC (2014) A survey of multiobjective evolutionary algorithms for data mining: part i. IEEE Trans Evol Comput 18(1):4–19
20. Bodenhofer U (2003) Genetic algorithms: theory and applications. Lecture notes, Fuzzy logic laboratorium Linz-Hagenberg, Winter
21. Sastry K, Goldberg DE, Kendall G (2014) Genetic algorithms. In: Search methodologies, pp 93–117
22. Goldberg DE, Holland JH (1988) Genetic algorithms and machine learning. Mach Learn 3(2):95–99
23. Baker JE (1987) Reducing bias and inefficiency in the selection algorithm. In: Proceedings of the second international conference on genetic algorithms, pp 14–21
24. Goldberg DE, Deb K (1991) A comparative analysis of selection schemes used in genetic algorithms. Found Genetic Algorithms 1:69–93
25. Schlierkamp-Voosen D, Mühlenbein H (1993) Predictive models for the breeder genetic algorithm. Evol Comput 1(1):25–49
26. Spears WM, De Jong KD (1995) On the virtues of parameterized uniform crossover. Technical report, Naval Research Lab Washington DC
27. Sivrikaya-Şerifoğlu F (1997) A new uniform order-based crossover operator for genetic algorithm applications to multi-component combinatorial optimization problems. Unpublished PhD dissertation, Boğaziçi University, Istanbul
28. Paul PV, Ramalingam A, Baskaran R, Dhavachelvan P, Vivekanandan K, Subramanian R (2014) A new population seeding technique for permutation-coded genetic algorithm: service transfer approach. J Comput Sci 5(2):277–297
29. Deng Y, Liu Y, Zhou D (2015) An improved genetic algorithm with initial population strategy for symmetric TSP. In: Mathematical problems in engineering
30. Hassanat AB, Prasath V, Abbadi MA, Abu-Qdari SA, Faris H (2018) An improved genetic algorithm with a new initialization mechanism based on regression techniques. Information 9(7):167
31. Kaya M (2011) The effects of two new crossover operators on genetic algorithm performance. Appl Soft Comput 11(1):881–890
32. Rafsanjani MK, Eskandari S (2011) A new combinational selection operator in genetic algorithm. AIP Conf Proc 1389:1082–1085 (**AIP**)
33. Rafsanjani MK, Eskandari S (2012) The effect of a new generation based sequential selection operator on the performance of genetic algorithm. Indian J Sci Technol 5(12):3758–3761
34. Hussain A, Muhammad YS (2020) Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator. Complex Intell Syst 6(1):1–14
35. Kaya Y, Uyar M, Tekın R (2011) A novel crossover operator for genetic algorithms: ring crossover. arXiv preprint arXiv:1105.0355
36. Semenkin E, Semenkina M (2012) Self-configuring genetic algorithm with modified uniform crossover operator. In: International conference in swarm intelligence, Springer, pp 414–421
37. Thakur M (2014) A new genetic algorithm for global optimization of multimodal continuous functions. J Comput Sci 5(2):298–311
38. Elsayed SM, Sarker RA, Essam DL (2014) A new genetic algorithm for solving optimization problems. Eng Appl Artif Intell 27:57–69
39. Osaba E, Onieva E, Carballedo R, Diaz F, Perallos A (2014) An adaptive multi-crossover population algorithm for solving routing problems. In: Nature inspired cooperative strategies for optimization (NICSO 2013), pp 113–124
40. Jin C, Li F, Tsang EC, Bulysheva L, Kataev MY (2017) A new compound arithmetic crossover-based genetic algorithm for constrained optimisation in enterprise systems. Enterpr Inf Syst 11(1):122–136
41. Demirci H, Ozcerit A, Ekiz H, Kutlu A (2015) Chaotic crossover operator on genetic algorithm. In: Proceedings of 2nd international conference on information technology
42. Alkafaween E (2018) Novel methods for enhancing the performance of genetic algorithms. CoRR https://arxiv.org/abs/1801.02827

43. Hassanat AB, Alkafaween E (2017) On enhancing genetic algorithms using new crossovers. Int J Comput Appl Technol 55(3):202–212
44. Xue Y, Zhu H, Liang J, Słowik A (2021) Adaptive crossover operator based multi-objective binary genetic algorithm for feature selection in classification. Knowl Based Syst 227:107218
45. Koohestani B (2020) A crossover operator for improving the efficiency of permutation-based genetic algorithms. Expert Syst Appl 151:113381
46. Manzoni L, Mariot L, Tuba E (2020) Balanced crossover operators in genetic algorithms. Swarm Evol Comput 54:100646
47. Viana MS, Morandin Junior O, Contreras RC (2020) A modified genetic algorithm with local search strategies and multi-crossover operator for job shop scheduling problem. Sensors 20(18):5440
48. Albayrak M, Allahverdi N (2011) Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms. Expert Syst Appl 38(3):1313–1320
49. Marung U, Theera-Umpon N, Auephanwiriyakul S (2016) Top-n recommender systems using genetic algorithm-based visual-clustering methods. Symmetry 8(7):54
50. Yuan Y, Wang W, Pang W (2021) A genetic algorithm with tree-structured mutation for hyperparameter optimisation of graph neural networks. In: 2021 IEEE congress on evolutionary computation (CEC), IEEE, pp 482–489
51. Haghrah A, Nekoui M, Nazari-Heris M, Mohammadi-ivatloo B (2021) An improved real-coded genetic algorithm with random walk based mutation for solving combined heat and power economic dispatch. J Ambient Intell Humaniz Comput 12(8):8561–8584
52. Alhijawi B, Kilani Y (2020) A collaborative filtering recommender system using genetic algorithm. Inf Process Manag 57(6):102310
53. Armagan A, Dunson DB, Lee J (2013) Generalized double pareto shrinkage. Stat Sin 23(1):119
54. Kumar M, Husian M, Upreti N, Gupta D (2010) Genetic algorithm: review and application. Int J Inf Technol Knowl Manag 2(2):451–454
55. Paulinas M, Ušinskas A (2007) A survey of genetic algorithms applications for image enhancement and segmentation. Inf Technol Control 36:3
56. Połap D (2020) An adaptive genetic algorithm as a supporting mechanism for microscopy image analysis in a cascade of convolution neural networks. Appl Soft Comput 97:106824
57. Sun Y, Xue B, Zhang M, Yen GG, Lv J (2020) Automatically designing CNN architectures using the genetic algorithm for image classification. IEEE Trans Cybern 50(9):3840–3854
58. Chen R, Yang B, Li S, Wang S (2020) A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. Comput Ind Eng 149:106778
59. Zhou Z, Li F, Zhu H, Xie H, Abawajy JH, Chowdhury MU (2020) An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. Neural Comput Appl 32(6):1531–1541
60. Maulik U, Bandyopadhyay S (2000) Genetic algorithm-based clustering technique. Pattern Recogn 33(9):1455–1465
61. Sheikh RH, Raghuwanshi MM, Jaiswal AN (2008) Genetic algorithm based clustering: a survey. In: 2008. ICETET'08. First international conference on emerging trends in engineering and technology, IEEE, pp 314–319
62. Mohammadrezapour O, Kisi O, Pourahmad F (2020) Fuzzy c-means and k-means clustering with genetic algorithm for identification of homogeneous regions of groundwater quality. Neural Comput Appl 32(8):3763–3775
63. Harman M, McMinn P, De Souza JT, Yoo S (2012) Search based software engineering: techniques, taxonomy, tutorial. In: Empirical software engineering and verification, pp 1–59
64. Srivastava PR, Kim T (2009) Application of genetic algorithm in software testing. Int J Softw Eng Appl 3(4):87–96
65. Harman M (2007) The current state and future of search based software engineering. In: 2007 Future of software engineering, IEEE Computer Society, pp 342–357
66. Ramkumar R, Mala G (2021) Non functional requirement based software architecture scheme with security requirement using hybrid group search optimization and genetic algorithm. J Ambient Intell Humaniz Comput 12(5):4863–4876
67. Alhijawi B, Awajan A (2021) Novel textual entailment technique for the Arabic language using genetic algorithm. Comput Speech Lang 68:101194
68. Iqbal F, Hashmi JM, Fung BC, Batool R, Khattak AM, Aleem S, Hung PC (2019) A hybrid framework for sentiment analysis using genetic algorithm based feature reduction. IEEE Access 7:14637–14652
69. Ar Y, Bostanci E (2016) A genetic algorithm solution to the collaborative filtering problem. Expert Syst Appl 61:122–128
70. Dwivedi P, Kant V, Bharadwaj KK (2018) Learning path recommendation based on modified variable length genetic algorithm. Educ Inf Technol 23(2):819–836
71. Hashemi S, Kiani S, Noroozi N, Moghaddam ME (2010) An image contrast enhancement method based on genetic algorithm. Pattern Recogn Lett 31(13):1816–1824
72. Daniel E, Anitha J (2015) Optimum green plane masking for the contrast enhancement of retinal images using enhanced genetic algorithm. Optik Int J Light Electron Opt 126(18):1726–1730
73. Deborah H, Arymurthy AM (2010) Image enhancement and image restoration for old document image using genetic algorithm. In: 2010 Second international conference on advances in computing, control and telecommunication technologies (ACT), IEEE, pp 108–112
74. Guo F, Peng H, Tang J (2016) Genetic algorithm-based parameter selection approach to single image defogging. Inf Process Lett 116(10):595–602
75. Wijayaningrum VN, Mahmudy WF (2016) Optimization of ship's route scheduling using genetic algorithm. Indones J Electr Eng Comput Sci 2(1):180–186
76. Xu Y, Li K, Hu J, Li K (2014) A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. Inf Sci 270:255–287
77. Faghihi V, Reinschmidt KF, Kang JH (2014) Construction scheduling using genetic algorithm based on building information model. Expert Syst Appl 41(16):7565–7578
78. Konar D, Bhattacharyya S, Sharma K, Sharma S, Pradhan SR (2017) An improved hybrid quantum-inspired genetic algorithm (Hqiga) for scheduling of real-time task in multiprocessor system. Appl Soft Comput 53:296–307
79. Keshanchi B, Souri A, Navimipour NJ (2017) An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing. J Syst Softw 124:1–21
80. Soundarya V, Kanimozhi U, Manjula D (2017) Recommendation system for criminal behavioral analysis on social network using genetic weighted k-means clustering. JCP 12(3):212–220
81. Wang Z, Yu X, Feng N, Wang Z (2014) An improved collaborative movie recommendation system using computational intelligence. J Vis Lang Comput 25(6):667–675
82. Georgiou O, Tsapatsoulis N (2010) Improving the scalability of recommender systems by clustering using genetic algorithms. In:

International conference on artificial neural networks, Springer, pp 442–449
83. El-Samak AF, Ashour W (2015) Optimization of traveling salesman problem using affinity propagation clustering and genetic algorithm. J Artif Intell Soft Comput Res 5(4):239–245
84. Rahman MA, Islam MZ (2014) A hybrid clustering technique combining a novel genetic algorithm with k-means. Knowl Based Syst 71:345–365
85. Shaw MKE (2015) K-means clustering with automatic determination of k using a multiobjective genetic algorithm with applications to microarray gene expression data. PhD thesis, San Diego State University
86. Lahari K, Murty MR, Satapathy SC (2015) Partition based clustering using genetic algorithm and teaching learning based optimization: performance analysis. In: Emerging ICT for bridging the future-proceedings of the 49th annual convention of the computer society of India CSI, vol 2, Springer, pp 191–200
87. Ahmed MA, Hermadi I (2008) Ga-based multiple paths test data generator. Comput Oper Res 35(10):3107–3124
88. Rao KK, Raju G, Nagaraj S (2012) Optimizing the software testing efficiency using genetic algorithm-implementation methodology. Softw Eng Technol 4(10):432–439
89. Räihä O (2010) A survey on search-based software design. Comput Sci Rev 4(4):203–249
90. Bhatia N (2016) A cluster adaptive genetic model for improving the recommender system. Imp J Interdiscip Res 2:8
91. Gupta A, Shivhare H, Sharma S (2015) Recommender system using fuzzy c-means clustering and genetic algorithm based weighted similarity measure. In: International conference on computer, communication and control (IC4), IEEE
92. Alahmadi DH, Zeng X-J (2015) Twitter-based recommender system to address cold-start: a genetic algorithm based trust modelling and probabilistic sentiment analysis. In: IEEE 27th international conference on tools with artificial intelligence (ICTAI), IEEE, pp 1045–1052
93. Verma A, Virk HK (2015) A hybrid recommender system using genetic algorithm and KNN approach. Int J Comput Sci Trends Technol (IJCST) 6(3):131–134
94. Verma A, Virk H (2015) A hybrid genre-based recommender system for movies using genetic algorithm and KNN approach. Int J Innov Eng Technol 5(4):48–55
95. Alhijawi B, Kilani Y (2016) Using genetic algorithms for measuring the similarity values between users in collaborative filtering recommender systems. In: 2016 IEEE/ACIS 15th international conference on computer and information science (ICIS), IEEE, pp 1–6
96. Xiao J, Luo M, Chen J-M, Li J-J (2015) An item based collaborative filtering system combined with genetic algorithms using rating behavior. In: International conference on intelligent computing, Springer, pp 453–460
97. Alhijawi B, Kilani Y, Alsarhan A (2020) Improving recommendation quality and performance of genetic-based recommender system. Int J Adv Intell Paradig (IJAIP) 10:1
98. Fong S, Ho Y, Hang Y (2008) Using genetic algorithm for hybrid modes of collaborative filtering in online recommenders. In: Eighth international conference on hybrid intelligent systems, HIS'08, IEEE, pp 174–179
99. Salehi M (2014) Latent feature based recommender system for learning materials using genetic algorithm. Inf Syst Telecommun 137
100. Athani M, Pathak N, Khan AU (2014) Dynamic music recommender system using genetic algorithm. Int J Eng Adv Technol 3(4):230–232
101. Zhang F, Chang H-y (2006) A collaborative filtering algorithm employing genetic clustering to ameliorate the scalability issue, IEEE, pp 331–338