Comput Syst Sci & Eng (2020) 1: 13–19
© 2020 CRL Publishing Ltd

International Journal of
Computer Systems
Science & Engineering

Developing an Adaptation Process for Real-Coded Genetic Algorithms

Ridvan Saraçoğlu*[*] and Ahmet Fatih Kazankaya[†]

Department of Electrical and Electronics Engineering, Van Yuzuncu Yil University, 65080, Turkey

The genetic algorithm (GA) is a metaheuristic method which simulates the life cycle and the survival of the fittest in the nature for solving optimization problems. This study aimed to develop enhanced operation by modifying the current GA. This development process includes an adaptation method that contains certain developments and adds a new process to the classic algorithm. Individuals of a population will be trialed to adapt to the current solution of the problem by taking them separately for each generation. With this adaptation method, it is more likely to get better results in a shorter time. Experimental results show that this new process accelerated the algorithm and a certain solution has been reached in fewer generations. In addition, better solutions were achieved, especially for a certain number of generations.

Keywords: Adaptive algorithms, Algorithm design and analysis, Genetic algorithms, Value coding genetic algorithms

1. INTRODUCTION

The genetic algorithm (GA) is an optimization and searching method, that belongs to an extended class of algorithms that are inspired by the natural selection process and are similar to the evolutionary phase [1]. GA is widely used to produce high quality genes for searching and optimization problems based on biological facts such as selection, crossover and mutation. In the GA, candidate individuals are expected to evolve the existing population to better genes for optimization problems [2]. Each candidate must have a number of features that can be mutated and modified. Evolutionary stage begins with a population that is generated randomly, then the fitness value of every single individual is calculated and continued as a new generation in each iteration [3]. Individuals with greater fitness values are chosen from the current population and the genes of individuals arealtered to a new form [4]. The generation that created currently is used in the next phases of the algorithm [5–7].

Once the fitness function has been identified, a gene population begins to be generated through repeated application of selection, evolution, crossover, and mutation stages [8, 9]. The population size entirely depends on the situation and the parameters. The first population is usually produced indiscriminately and all possible gene variations are allowed in this scenario [10]. In each generation, a part of the current population is chosen to produce the next generation. Genes are lined up from low to high as for that their fitness values, and genes that are marked with higher fitness values are most likely to be combed out depending on the parameters. Fitness functions that evaluate the suitability of each gene are responsible for selecting the best genes [4, 11].

Depending on the problem formula, the function is defined for the simulated gene while also measuring the quality of the gene [6, 12]. Next, new generation populations are produced from the genes that were selected by the sequential combination of crossover and mutation [7]. While reaching reasonable results for the problem being studied, it is very important to set the parameters of mutation, crossover factor and population size [11]. A pair of parents are selected from the population for new genes to be produced [13]. The crossover method generates a child gene that shares most of the qualifications of the parents [4]. New parents are chosen in order to create new offspring and the process sustains until a new solution size is found [14–16].

[*]ridvansaracoglu@yyu.edu.tr
[†]afkazankayaa@hotmail.com

As a result of these processes, new generation chromosomes are formed which are completely different from the first generation [17, 18]. It is expected that the average value of the fitness value of the population will increase by these processes because the genes with the best fitness values and other genes with slightly less fitness value are selected for the crossover process [19]. The main reason to keep these genes with less fitness value is to increase the genetic diversity in both the genetic pool and future generations [20, 21]. New offspring and some randomly selected genes are subjected to low-probability mutations. This allows some bits to be changed in their bit string [17, 18, 22]. The mutation process is often used to retain variety in the population and prevent early convergence [23–25]. The algorithm ends when either the maximum population is generated or the aimed fitness value of the population is reached [26].

Attempts have been made for years ed to develop and expand the GA by considering various factors to reach the desired solution in a shorter and more efficient way. These may vary depending on the purpose of the investigator; purposes such as; optimization design [24, 27, 28], convenience function and range [29], selection method [30, 31], population type and size [32], iteration number [33], crossing and mutation values [34], elite individual coefficient, stop criteria [25, 35] and gene pool [36] can be important factors.

In this study, a new adaptation process will be added between the crossover and mutation process which are the two main genetic stages of the GA. In this stage, firstly the best individual in the algorithm so far is determined as the donor individual. The individual that has got the best fitness value among the population (donor) is determined and cloned on the randomly determined ones. Thus, the goal of the algorithm is to reach generations much earlier and with much better individuals.

The rest of the article details and gives specific information about the GA. In Chapter 2, selection, crossover, and mutation processes are discussed in detail. In Chapter 3, detailed information about the process and the placement of the adaptation phase is given. In Chapter 4, the experimental results that were obtained with the selected test functions and comments on these results are given. Finally, Chapter 5 contains the conclusions of the article.

2. BACKGROUND

In this section, the basic elements of the GA which are the repetitive main stages of the traditional genetic algorithm (TGA) such as selection, crossover and mutation are given [37]. After examining these phases, details about the proposed adaptation phase to be added to the algorithm will be given.

The GA is a method based on artificial intelligence, survival of the fittest and evolutionary biology theory that used to find optimized solutions for problems and to make extensive searches among large data sets [6, 38–40].

2.1 Encoding and Initial Population

Encoding genes has great importance, when it is started to solve problems using GA. Encoding depends on the problem, and it is necessary to choose an encoding technique suitable for the problem. In this chapter, the types of encoding techniques will be introduced.

Binary encoding is one of the most popular techniques since the very first studies about GA were used with this encoding. In binary encoding, a gene has a combination of a string of bits, 0 or 1. The binary encoding technique gives genes many combinations. From another point of view, this encoding technique is away being natural for many problems, and therefore, there should be some enlistments that need to be done after mutation and crossover phases [41].

Second, permutation encoding, which can be used kind of problems that include the traveling salesman problem or the task ordering problem. In this method, every chromosome is a combination of numbers. Permutation encoding has benefits if problem is an ordering one. This technique has mutation and crossover corrections like the other techniques with the difference of leaving the chromosome consistent [7].

Next, direct value encoding which can be used with real value numbers. Using this technique brings some difficulties to the algorithm. In this type of encoding, every has a combination of a string that can be almost everything according to the problem [42]. Value encoding is very effective for certain problems, and this encoding is often essential to develop a new mutation and crossover algorithm in order to make whole algorithm work correctly [6].

Tree encoding is mainly used for progressing a program. In this technique\ every gene is a combination of a tree scheme of articles. Tree encoding is useful for progressing or developing programs like LISP [33].

In the TGA, a population containing as many individuals as the determined size is created. Population individuals contain completely random values in the initial population size. The number of genes of the current parameter size is created according to the feature of the problem. The fitness values can be calculated according to random values assigned to genes and determined fitness function [6, 26].

2.2 Selection

In principle, an individual population randomly selected from the search field offers candidate solutions to the problem. Then, these individuals will be crossed-over and mutated into new offspring. The new population is finally created by a procedure between parents and offspring. This process replicates until a aimed circumstanced is reached [43]

In Roulette Wheel Selection, the fitness function value is proportional to the probability of selection. In Rank Selection, the order of their fitness value degree is proportional to the probability of selection. [44]. In the Tournament Selection Method, a certain number of individuals are indiscriminately picked and entered into the tournament according to their fitness values. For example, two individuals would be indiscriminately elected from the population as the 4th and 2nd individuals. One of these individuals will be selected when they are entered into the tournament among themselves: such as the 2nd individual [45].

Those selection methods have different properties. For example, the roulette wheel selection method will reduce the diversity of the gene pool according to other selection features.

Because individuals have the possibility of being directly selected according to their suitability, the number of the best individuals in the population will be higher. This event increases the risk of tripping to the best local gene [45].

2.3 Crossover

Crossover, a method for recombination in GAs, is a genetic phase that is used to produce and integrate the genetic combination of two parents into new two offsprings. It is aa efficient way of producing a new offspring from a stochastically available population and resembles the gateway that occurs during duplicating in biology [44]. Various algorithms in the evolutionary calculation can involve various data to warehouse genetic combinations, and each gene can be reassembled with different attendant that belongs to crossover phase [46].

The crossover method is usually used to spawn two offspring's chromosomes from two selected chromosomes. In this operator, the algorithm first selects the first two parental individuals and then, by subjecting these two individuals to the determined crossover process, creates two children according to the specific gene variation. New individuals will continue to be created according to the number of parents and populations which were determined earlier [47].

One of the most popular methods for crossover is single-point crossover. A random point, which is called the 'crossover point', is determined on the chromosomes of both parents. This occurs in two offspring transporting the same gen combination from both parents [23].

Another set of popular ways to constrain a crossover operation are the two-point and k-point crossover methods. In two-point crossover, two different points are indiscriminately chosen from the main gene combination. Bits that remain between these two points are switched between the main chromosomes. This operation is almost the same as implementing two single-point crossovers with particular combination of crossover points. This method collects the k-point crossover that can be generalized to the k crossover points [23].

There is also the uniform crossover method. In uniform crossover, one bit from each genome is selected independently from two parents. The uniform crossover replaces the individual bits, not the whole sequence [48]. This means that there are no hesitations for the two bits which are close to each other in the sequence to be taken together [23]. Frequently, each bit is selected from an equal probability parent. Sometimes other proportions are used that cause offspring to get more inheritance from one than another parent.

2.4 Mutation

The main object of this operation is to provide gene diversity for present and future generations. Mutation changes gene values from the beginning phase of a chromosome. In the mutation, the gene combination of individuals may be completely different from the previous combination. Therefore, GA can achieve a better solution using this operator. It has the exact same coefficient as the crossover method. If that coefficient is set to a very high value, the results are expected to convert to a completely different random search [44, 49].

The mutation rate is used to hold whether to change the value of one or more genes of an individual. The algorithm will produce a indiscriminate value in the [0,1] range, which will be compared to this predetermined mutation rate. In cases where the generated random value is greater than the mutation rate, no changes will be made to that gene of the individual. In cases where the mutation rate is greater than or equal to the random number, a new value will be assigned according to the current gene variation [44, 49].

First, as one of the most common mutation methods, Flip Mutation is predicated on an engendered mutation chromosome, flipping scarcely involves transmuting 0 to1 and 1 to 0. The mutation phase then occurs optionally by taking pair of parents into consideration [26]. This method is jointly taken place with binary encoding. Moreover, other mutation methods, such as Insert Mutation, Scramble Mutation, Swap Mutation, Interchanging Mutation, and Uniform Mutation can be given [50].

2.5 Termination

The selection, crossover and mutation procedures are executed in the loop. A termination criterion is required in order not to continue the process forever. Some of the basic termination criteria are the Maximum generation, the Optimization Target and the convergence of the population [51].

The maximum generation method is the limit on the generation number. In this method, the number of a generation is determined and the algorithm is stopped when aimed generation number is attained.

When target value of the objective function is reached, the algorithm is stopped. In the convergence method, improvement in the values of the new generation may indicate that no further improvement is expected. This means the termination of the algorithm.

3. PROPOSED ADAPTATION PROCESS

The GA combines two different approaches when looking for the optimum solution to a problem. The first is crossover, which enables the local optimum to be found faster during the search. New individuals are introduced with the help of a mixture of existing solutions. Another approach is mutation, which adds diversity to existing solutions. In this way, the algorithm is saved at a local minimum or maximum.

The GA attempts to create a model of the natural evolutionary process. Crossover assumes that an individual belongs to the new generation. Unusual changes that occur at the end of a generation are shown by mutation. These changes are called genotype changes. Survivors of mutated individuals increase diversity. However, there are also Phenotype modifications due to environmental impact on individuals. This change is a direct adaptation of the individual to the environment. These changes are not transferred to new generations. However, it provides the formation of more harmonious individuals. The likelihood

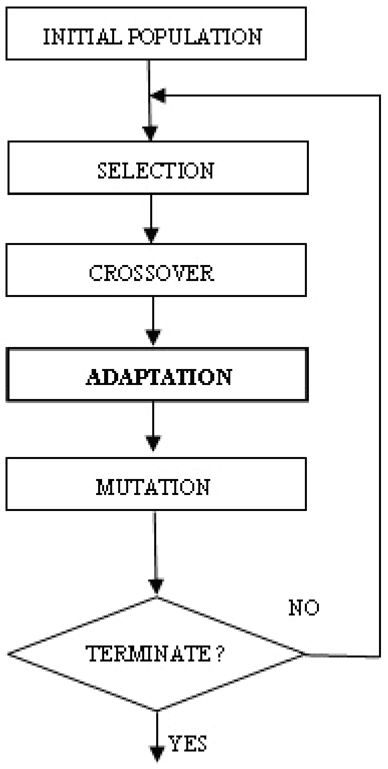

Figure 1 Simplified flow chart of proposed Genetic Algorithm with Adaptation.

of survival of these individuals also increases. Environmental conditions make physiological or morphological changes within a shorter period than the life of the individual [52]. For example, there are several studies on the effects of environmental conditions on size changes in the organ systems of adult organisms [53, 54].

The conclusion from this point is that individuals within the population are crossover and mutated. In addition to these processes, individuals adapt to their environment. This phenomenon can be considered as an adaptation of the individual to the environment. In other words, crossover and mutation are indispensable processes of GA, but they are insufficient to reflect the natural process. The purpose of this study is to eliminate this deficiency. Our main goal is to design the adaptation process to secure that the genetic process in nature is transformed into an optimization algorithm.

The essence of adaptation is that the individual becomes more adaptable to the environment. However, in the case of GA processes, this is measured and evaluated by the conformity function. Thus, if a gene is more compatible with environmental conditions, it can be listed as what should be done. The best individual of the previous and current populations is whichever is the most compatible with the environment. The individual to be adapted must naturally be similar to this best person. In the continuation of the text, this will be called the best individual donor.

The proposed adaptation process locates between crossover and mutation processes and is shown in Fig. 1. In here, it is given a simple flow chart of the GA that is added the adaptation process.

As in the mutation process, the adaptation process is also based on the gene. The first thing to do is to determine which genes should be adapted. Generally, this is a standard process independent of the encoding method. As with mutation and crossover operations, an operator has been proposed for the adaptation process. This parameter, called the adaptation rate, is a real number ranging from 0 to 1. The decision is made based on whether or not to adapt the genes. An indiscriminate number will be created and if the number is smaller than the adaptation rate, that gene will be modified.

The second is how to change a gene to adapt. This is described in detail below through the adaptation process designed for different encoding methods.

3.1 Classical Adaptation

In Direct Value Encoding, the genes are integers or real numbers. The value of the gene to be adapted is assigned the value of the same gene of the donor. This method is called classical adaptation.

3.2 K-Individual Adaptation

In the K-individual Adaptation method, the adaptive gene will be updated using some individuals. These individuals are the best individuals in the whole algorithm. In Direct Value Encoding, since the genes are integers or real numbers, the average of these values creates the value to be used in the update.

4. EXPERIMENTS AND RESULTS

In this section, the effect of the proposed adaptation process on GA was tested using five different benchmark functions. Four

Table 1 Parameters of GA.

Parameter	TGA	GA with Adaptation
Encoding	Value	Value
Population size	50	50
Selection	Tournament	Tournament
Crossover rate	0.8	0.8
Adaptation rate	None	0.1
Mutation rate	0.1	0.1
Max Generation	400	400

Table 2 Generation number results of optimal solutions for test function.

Test Function	TGA		GA with Adaptation	
	Mean	STD	Mean	STD
Ackley	179.7	93.69	57.6	43.21
Eggholder	352.1	24.42	252	74.27
Holder Table	203.1	98.38	139.3	104.56
Rastrigin	240.2	113.32	80.4	33.26
Bohachevsky	234.9	124.43	92.5	81.76

Table 3 Runtime results of optimal solution (seconds) for test functions.

Test Function	TGA		GA with Adaptation	
	Mean	STD	Mean	STD
Ackley	0.0628	0.0069	0.0519	0.0064
Eggholder	0.0762	0.0104	0.0694	0.0057
Holder Table	0.0612	0.0069	0.0588	0.0103
Rastrigin	0.0666	0.0209	0.0537	0.0129
Bohachevsky	0.0715	0.0266	0.0523	0.0083

Table 4 Results of test functions for 300 generations of GA.

Test Function	TGA		GA with Adaptation	
	Mean	STD	Mean	STD
Ackley	1.88E-2	2.6E-2	1.14E-3	2.8E-3
Eggholder	-940.69	0.44	-959.46	0.12
Holder Table	-18.57	0.34	-19.21	0.03
Rastrigin	6.91E-4	1.2E-3	2.19E-7	6.9E-7
Bohachevsky	5.63E-3	1.1E-2	1.82E-5	3.9E-5

of these functions were multimodal (Ackley, Eggholder, Holdertable, Rastrigin) and the last was the unimodal (Bohachevsky) test function. In the test functions, the dimension was selected as 3.

To verify the productivity of the proposed adaptation process, TGA and the GA that includes the proposed adaptation process were run with the same parameters. The selected population size was 50 for both algorithms. The method used for the adaptation process was classical adaptation. The algorithms were run until the test functions reached the optimal solution. The statistical results of each test function were determined by 30 independent runs. The parameter list used for GA is shown in Table 1.

The hardware used for the application was an Intel (R) Core (TM) i5-6500 CPU 3.20 GHz and the software used was NetBeans IDE 8.2.

As shown in Table 2, the proposed adaptation process accelerated the algorithm significantly while the number of generations needed for achieving the optimum solution has decreased considerably. This applies to both the multimodal test functions and the unimodal test function and reveals the necessity of the adaptation process.

It is obvious that the adaptation process brings additional complexity to GA. It draws attention to the impact of the proposed process on the GA runtime. In Table 3, the effect of the method in terms of working time was examined. This table presents the CPU time in seconds for each benchmark function. Although adding complexity, the working time decreases due to the decreasing generation number needed for reaching the optimum solution.

The performance of methods with constant generation number is compared in Table 4. As it is understood from this point of view, the adaptation process decreases the result values of the test functions and this means that the success is increased.

The main purpose of this section is to compare the proposed method with other GA studies in the literature. Previous studies are commonly based on the development and improvement of the traditional GA method. In addition, each study has its own set of parameters. In order to get a proper comparison, the parameters

Table 5 Comparison of proposed method with other GA based methods.

Methods	Dimension	Population size	Max Generation	Mean	STD
QGA-MPC[55]	10	100	500	27.378	2.068
GA with Adaptation	10	100	500	1.46E-2	2.39E-2
QGA-MPC[55]	30	100	1500	232.408	10.54
GA with Adaptation	30	100	1500	1.56E-5	8.09E-5
QGA-MPC[55]	50	100	2500	517.172	13.224
GA with Adaptation	50	100	2500	1.08E-5	4.91E-6
MLEO-C[56]	30	200	1000	1.98E-2	4.73E-2
GA with Adaptation	30	200	1000	4.85E-7	1.36E-6
FMD-U&N[57]	25	400	250	2.6	-
GA with Adaptation	25	400	250	6.3E-3	6.9E-3

must be the same. Thus, the parameter that is used in the previous studies were also used for the method that we proposed. The comparison which has been made by considering this necessity is given in Table 5.

The selected test function for comparison is the Rastrigin function. Mean and standard deviation values are given by considering the results of 30 independent operations for each value.

As can be clearly seen from Table 5, the proposed method is more successful than other GA-based methods when the same set of parameters are considered.

5. CONCLUSION

In this study, a new additional process has been proposed for GA. With this process, called adaptation, the aim is to adapt the individuals in the population to the best solution of the algorithm up to that step. In this way, natural selection, crossover, mutation and adaptation are also included in the GA. As a result of the evaluation with various test functions, the algorithm has developed clearly. Experimental results showed that this new process accelerated the algorithm and a certain solution was reached in fewer generations. In addition, better solutions were achieved especially for a fixed number of generations.

The global optimum was achieved with both fewer generations and less time. This situation supports the idea that an adaptation process is a natural and necessary part of evolutionary algorithms. In the literature, more efficient optimization models generally use hybrid structures. It is thought that the adaptation process we proposed could increase the success of all hybrid systems including GA. In addition, other evolutionary algorithms (differential evolution, etc.) can be developed through an adaptation process.

Genetic processes, such as crossover, mutation, and other coding methods can show differences in their methods. Although the adaptation process is proposed only for value coding, an adaptation process will also be developed for other coding methods such as binary and permutation coding.

Conflict of Interest

The authors declare that they have no conflict of interest.

REFERENCES

1. Mitchell M (1996). An Introduction to Genetic Algorithms. MIT Press, Cambridge.
2. Whitley D (1994). A Genetic Algorithm Tutorial by Darrell Whitley. Stat Comput 65–85.
3. Koza JR (1992). Genetic programming: On the programming of computers by means of natural selection.
4. Hu M, Huang GH, Sun W, et al. (2014). Multi-objective ecological reservoir operation based on water quality response models and improved genetic algorithm: A case study in Three Gorges Reservoir China. Engineering Application of Artificial Intelligence, 36: 332–346.
5. Banzhaf W, Nordin P, Keller RE, Francone FD (1998). Genetic programming: an introduction.
6. Mc Call J (2005). Genetic algorithms for modelling and optimization. Journal of Computational and Applied Mathematics, 184: 205–222.
7. Xing LN, Chen YW, Yang KW, et al. (2008). A hybrid approach combining an improved genetic algorithm and optimization strategies for the asymmetric traveling salesman problem. Engineering Application of Artificial Intelligence, 21: 1370–1380.
8. Sadeghi E, Shima S., Sleno L, Sabally K. (2016). Biotransformation of polyphenols in a dynamic multistage gastrointestinal model. Food Chemistry, 204: 453–462.
9. Shir OM (2012), Niching in evolutionary algorithms", In: Handbook of Natural Computing. 1036–1069.
10. Akbari M., Rashidi H, Alizadeh SH (2017). An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems. Engineering Application of Artificial Intelligence, 61: 35–46.
11. Goldberg DE (1989). Genetic Algorithms in Search, Optimization, and Machine Learning.
12. Taherdangkoo M, Yazdi M., Bagheri M (2012). A powerful and efficient evolutionary optimization algorithm based on stem cells algorithm for data clustering. Open Computer Science, 2.
13. Echegoyen C, Mendiburu A, Santana R., Lozano JA (2013). On the taxonomy of optimization problems under estimation of distribution algorithms. Evolutionary Computation, 21: 471–495.
14. Cha SH, Tappert C (2009). A Genetic Algorithm for Constructing Compact Binary Decision Trees. Journal of Pattern Recognition Research, 4(1): 1–13.
15. Guo P, Wang X, Han Y (2010), The enhanced genetic algorithms for the optimization design. In: Proceedings -3rd International Conference on Biomedical Engineering and Informatics, BMEI 2010, 990–2994.
16. Poon PW, Carter JN (1995). Genetic algorithm crossover operators for ordering applications. Computers & Operations Research, 22: 135–147.

17. Harik GR (1997). Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms. Doctoral dissertation, University of Michigan.
18. Zhang J, Chung HSH, Lo WL (2007). Clustering-based adaptive crossover and mutation probabilities for genetic algorithms. IEEE Transactions on Evolutionary Computation, 11: 326–335.
19. Janikow CZ, Michalewicz Z (2002). A specialized genetic algorithm for numerical optimization problems. 798–804.
20. Patrascu M, Stancu AF, Pop F (2014)., HELGA: a heterogeneous encoding lifelike genetic algorithm for population evolution modeling and simulation, Soft Computing, 18: 2565–2576.
21. Baluja S, Caruana R (2014). Removing the Genetics from the Standard Genetic Algorithm. In: Machine Learning Proceedings, 38–46.
22. Chen SY, Zheng F, Wu SQ, Zhu ZZ (2017). An improved genetic algorithm for crystal structure prediction. Current Applied Physics, 17: 454–460.
23. Holland JH (1975). Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.
24. Hu XB, Wu SF, Jiang J (2004). On-line free-flight path optimization based on improved genetic algorithms. Engineering Application of Artificial Intelligence, 17: 897–907.
25. Király A, Abonyi J (2015). Redesign of the supply of mobile mechanics based on a novel genetic optimization algorithm using Google Maps API. Engineering Application of Artificial Intelligence, 38: 122–130.
26. Gürbüz A (2010). Improved Genetic Algorithm. Doctoral dissertation, Marmara University.
27. Bakirtzis AG, Biskas PN, Zoumas CE, Petridis V (2002). Optimal power flow by enhanced genetic algorithm. IEEE Transaction on Power Systems, 17: 229–236.
28. Shen Y (2018). Improved chaos genetic algorithm based state of charge determination for lithium batteries in electric vehicles. Energy, 152: 576–585.
29. Zheng Z, Zheng Z (2018). Towards an improved heuristic genetic algorithm for static content delivery in cloud storage. Computers and Electrical Engineering, 69: 422–434.
30. K. Wang K,Salhi A, Fraga ES (2003). Cluster analysis and visualisation enhanced genetic algorithm. Computer Aided Chemical Engineering, 15: 642–647.
31. Alam T, Raza Z (2018). Quantum genetic algorithm based scheduler for batch of precedence constrained jobs on heterogeneous computing systems. Journal of Systems Software, 135: 126–142.
32. Elsayed SM, Sarker RA, Essam DL (2011), Improved genetic algorithm for constrained optimization. In: Proceedings - ICCES'2011 International Conference on Computer Engineering and Systems. 111–115.
33. Shrestha A, Mahmood A (2016). Improving Genetic Algorithm with Fine-Tuned Crossover and Scaled Architecture. Journal of Mathematics, 2016: 1–10.
34. Huynh NT, Huang YC, Chien CF (2018). A hybrid genetic algorithm with 2D encoding for the scheduling of rehabilitation patients. Computers and Industrial Engineering, 125: 221–231.
35. Xu Y, Li K, Hu J, Li K (2014). A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. Information Sciences, 270: 255–287.
36. Yang HJ, Hu X (2016). Wavelet neural network with improved genetic algorithm for traffic flow time series prediction. Optik, 127: 8103–8110.
37. Budin L, Golub M, Budin A (1996). Traditional techniques of genetic algorithms applied to floating-point chromosome representations. Proceedings of the 41st Annual Conference KoREMA, Zagreb, Hrvatska, 93-96.
38. Ting CK (2005). On the mean convergence time of multi-parent genetic algorithms without selection. In: Lecture Notes in Computer Science, 403–412.
39. Fraser A (2016). Simulation of Genetic Systems by Automatic Digital Computers VI. Epistasis. Australian Journal of Biological Sciences, 13: 150.
40. Bies RR, Muldoon MF, Pollock BG, et al. (2006). A genetic algorithm-based, hybrid machine learning approach to model selection. Journal of Pharmacokinet Pharmacodyn, 33: 195–221.
41. Samanta D (2016). Encoding Techniques in Genetic Algorithms. Indian Institute of Technology Kharagpu.
42. Park CH, Lee WI, Han WS,VautrinA (2008). Improved genetic algorithm for multidisciplinary optimization of composite laminates. Computers and Structures, 86: 1894–1903.
43. Fogel DB (2005). Evolutionary Computation: Toward a New Philosophy of Machine Intelligence: Third Edition.
44. Miller BL, Goldberg DE (1996). Genetic algorithms, selection schemes, and the varying effects of noise. Evolutionary Computation, 4: 113–131.
45. Jebari K, Madiafi M (2013). Selection Methods for Genetic Algorithms. International Journal of Emerging Sciences, 3: 333–344.
46. Tian Z, Yan Y, Hong Y, et al. (2018). Improved genetic algorithm for optimization design of a three-dimensional braided composite joint. Composite Structure, 206: 668–680.
47. Michalewicz Z (1996). Genetic algorithms + data structures = evolution programs. Springer-Verlag Berlin Heidelberg.
48. Eshelman LJ (1991). "The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. Foundations of Genetic Algorithm, 1: 265–283.
49. Srinivas M, Patnaik LM (1994). Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms. IEEE Transactions on Systems, Man, and Cybernetics, 24: 656–667.
50. Soni N, Kumar T (2014). Study of Various Mutation Operators in Genetic Algorithms. International Journal of Computer Science and Information Technologies, 5: 4519–4521.
51. Safe M, Carballido J, Ponzoni I, Brignole N (2004), "On Stopping Criteria for Genetic Algorithms. In: Bazzan A.L.C., Labidi S. (eds) Advances in Artificial Intelligence – SBIA 2004, Lecture Notes in Computer Science, 3171.
52. Piersma T, Drent J (2003). Phenotypic flexibility and the evolution of organismal design. Trends in Ecology & Evolution,18: 228–233.
53. Hammond KA, Szewczak J, Krol E (2001). Effects of altitude and temperature on organ phenotypic plasticity along an altitudinal gradient. Journal of Experimental Biology, 204: 1991–2000.
54. Starck JM, Beese K (2002). Structural flexibility of the small intestine and liver of garter snakes in response to feeding and fasting. Journal of Experimental Biology, 205: 1377–1388.
55. Khuat TT, Le MH (2017). A genetic algorithm with multi-parent crossover using quaternion representation for numerical function optimization. Applied Intelligence, 46: 810-826.
56. Akbari R, Ziarati K (2011). A multilevel evolutionary algorithm for optimizing numerical functions. International Journal of Industrial Engineering Computations, 2: 419-430.
57. Garcia-Martinez C, Lozano M, Herrera F, Molina D, Sanchez AM (2008). Global and local real-coded genetic algorithms. European Journal of Operational Research, 185: 1088–1113.