Journal of Computing and Security

Research Article, 2022, Volume 9, Number 1 (pp. 45–56)

http://www.jcomsec.org

Gaussian Linear Regression Crossover for Genetic Algorithms

Pezhman Gholamnezhad [a,*]

[a] *Shahid Sattari Aeronautical University of Science and Technology, Tehran, Iran.*

ARTICLE INFO.

Article history:
Received: 30 October 2021
Revised: 22 April 2022
Accepted: 22 May 2022
Published Online: 29 June 2022

Keywords:
Genetic Algorithm, Crossover, Simulated Binary Crossover (SBX), Gaussian Process, Bayesian Linear Regression.

ABSTRACT

In the simulated binary crossover, offspring are generated from parents with a coefficient of variation and uses a probability distribution function for the coefficient and there is a linear relationship between parents and offspring. Most existing methods of crossover operators generate offspring on the solution on the decision space during the search and so far, no suggestion has been proposed on making a regression model for generating the offspring on the objective space. In this paper, a Gaussian linear regression crossover has been proposed. The idea is to apply linear regression to model a relationship between parents and offspring in crossover operations through the Gaussian process. The reason for using this process is that the probability distribution of the simulated binary operator is based on the parent in the mating pool on decision space, while the probability distribution of the proposed method is on objective space in the mating pool. To optimize problems on the combinatorial sets, the proposed method is applied. The performance of the proposed algorithm was tested on Computational Expensive Optimization benchmark tests and indicates that the proposed operator is a competitive and promising approach.

1 Introduction

1.1 Background

In artificial intelligence, a genetic algorithm (GA) is described as a metaheuristic, which is the mechanism of natural selection and genetic operators such as crossover, mutation, and reproduction, to guide the search space. In GA, the decision variables or input parameters of the problem are converted to solution strings of a finite length. Genetic algorithms start to search through a series of individuals which is called a population which is created at random. Genetic operators take global optimum solutions based on the solutions in the current populations which are reproduction, crossover, and mutation. Then newly generated individuals are replaced with the old population and the evaluation process goes forward until certain termination criteria are satisfied[1]. The performances of GAs depend partly on the genetic operator or selection strategy[2].

The problem of selection is a significant role in resolving precocious and slow convergence. Also, lower diversity in a population is a challenge to locally exploit the solutions[3].

In a genetic algorithm, the crossover also called recombination, is the most important operator and generates offspring to explore a wider area of the solution space. The crossover creates stochastically new solutions from an existing population and more than one parent is selected from the mating pool, and

* Corresponding author.

Email address: pezhman.gholamnezhad@gmail.com (P. Gholamnezhad)

https://dx.doi.org/10.22108/JCS.2022.131269.1084
ISSN: 2322-4460 © Research Article, 2022 JComSec. All rights reserved.

one or more offspring are generated. Typical data structures that can be merged in crossover operators are bit arrays, vectors of real numbers, or trees.

1.2 Motivation and Contributions

The basic idea in these crossovers is the replacement of genes in different ways. In simulated binary crossover (SBX) the offspring are generated from parents as follows:

$$\begin{cases} x_i^{(1,t+1)} = 0.5\left[(1+\beta_{qi})\, x_i^{(1,t)} + (1-\beta_{qi})\, x_i^{(2,t)}\right] \\ x_i^{(2,t+1)} = 0.5\left[(1-\beta_{qi})\, x_i^{(1,t)} + (1+\beta_{qi})\, x_i^{(2,t)}\right] \end{cases} \tag{1}$$

Which $\beta_q i$ is a probability distribution and calculated as follows:

$$\beta_{qi} = \begin{cases} (2u_i)^{\frac{1}{\eta_c+1}}, & \text{if } u_i \le 0.5 \\ \left(\frac{1}{2(1-u_i)}\right)^{\frac{1}{\eta_c+1}}, & \text{otherwise} \end{cases} \tag{2}$$

and u_i is a random number which $u_i \in [0,1)$. The offspring are generated with a coefficient of variation from parents. There is a linear relationship between parents and offspring: $x_i^{(t+1)} = W_i x_i^t$ which W_i and x_i^t are independent variables and $x_i^{(t+1)}$ is a dependent variable. Choosing the right coefficient of W_i can improve the diversity and convergence of offspring, while in simulated binary crossover (SBX), some offspring that may be efficient is not created. This is a challenge when there is local objective space, leading to poor solutions. In addition, the SBX procedure eliminates the chance of applying efficient decision variables, to the objective spaces. Further, this procedure cannot specify which decision variables are best efficient to assign to the objective functions. But so far, linear regression has not been used to model a relationship between parents and offspring in crossover operations. In this paper, we concentrate on the Bayesian linear regression to generate offspring from parents in the crossover operator. In Bayesian linear regression, there are $D = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. The "y" is modeled using the independent random variables and the distribution of "y" is a Gaussian process.

In probability and statics, the random variable X converts the sample space members into real numbers, the event space members into the Burrell B set of real numbers, and the event probability function into the probability random variable as illustrated in Figure 1:

In Figure 1, the probability space based on the event is converted to probability space based on a random variable. If X is a random variable, the probability

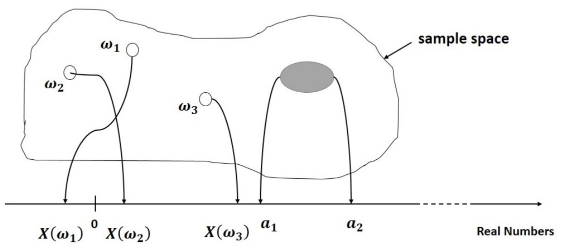

Figure 1. The Concept of a Random Variable.

distribution function of the random variable X is a function that takes real numbers between 0 and 1and is defined as follows:

$$F_X(y) = P(X \le y); y \in R \tag{3}$$

Which $F_X(y)$ is the value of the distribution function of the random variable X at the point y. Thus, this function expresses the probability value for the random variable X to the point y.

A probability density function (PDF) or density of a continuous random variable is a function whose it's integral in any given interval is equal to the probability of the random variable being in that interval. The parametric density estimation approaches are the estimative approach and predictive or Bayesian approach or nonparametric methods (Kernel regression (Nadaraya-Watson estimator), GP regression).

Gaussian processes are a non-parametric method. Parametric approaches distil knowledge about the training data into a set of numbers. For linear regression, this is just two numbers, the slope, and the intercept, whereas other approaches like neural networks may have 10s of millions. Being a Bayesian method, Gaussian Process makes predictions with uncertainty. In addition, the Gaussian processes model is a probabilistic supervised machine-learning framework that has been widely used for regression and classification tasks. A Gaussian processes regression (GPR) model can make predictions incorporating prior knowledge (kernels) and provide uncertainty measures over predictions. So, a new approach is proposed: Gaussian Linear Regression crossover (GLR-C) which a Gaussian process is used for the normal linear regression model.

The proposed method aims to generate more diverse offspring even in a poor mating pool to enhance the performance of diversity and convergence which is attained through Gaussian Linear Regression crossover (GLR-C).

The rest of this paper is formed as follows: The next section is related works. The problem statement is presented in Section 3 and briefly is reviewed some background necessary in the paper. The proposed algorithm is described in Section 3. Section 4 illustrates the experimental analysis of applying algorithm, results, and, computational complexity. Finally, Conclusions and future work are given in Section 5.

2 Related Work

In the last three decades, many types of crossover have been suggested. There are three main classifications of crossover[4]: standard crossover, binary crossover, and application-dependent crossover.

In the standard crossover, there are ten types of crossover operators: 1. 1-point cross over: two parents in a single point are broken randomly and then the parents at broken points are combined into a cross to create the offspring. 2. K-point crossover: two parents are broken randomly in k points and then two offspring are generated by the combination of parents at k points broken. 3. Shuffle crossover: this operator shuffles the genes in two parents randomly and then applies a 1-point crossover method to generate the offspring. 4. Reduced surrogate crossover: this operator compares the genes of two parents and if parents were different in more than 1 gene, it has listed all crossover points and then randomly selects one broken point and performs 1-points. 5. Uniform crossover: In this operator, a random real number determines that the first offspring selects the ith gene from the first or second parent. 6. Average crossover (AX): This operator generates one offspring from two parents by averaging from two parents. 7. Discrete crossover: This operator generates one offspring from two parents and a random real number determines which genes of parents must be selected. 8. Flat crossover: This operator performs the same as discrete cross over and the real number which is randomized, is selected, which is either min or max from genes in parents. 9. Heuristic crossover: This operator generates one offspring from two parents as below: $x_i(t+1) = x_i(t) + \alpha(y_i(t) - x_i(t))$. The parameter α is a uniform random real number. 10. Statics-based adaptive non-uniform crossover (SANUX): This operator uses information convergence from the gene locus to direct the crossover by regulating the swapping probability of each locus.

In the binary crossover, there are nine types of crossover operators: 1. Random Respectful Crossover (RSC): In this operator, two offspring are generated based on the likeness vector of two parents. 2. Masked crossover (MX): This operator applies a mask vector to specify which bits of parents are inherited by offspring. 3. 1-Bit Adaption Crossover (1BX): In this operator, the last bit of the solution vector determines which of these two crossover operators should be used. 4. Multivariate Crossover (MC): This operator divides the length of the parent into q substrings. Then a random value is selected for each q substring and according to the value, the genes of parents are copied into offspring. 5. Homologous Crossover (HX): In this operator, the strings of bits, which are at least of a certain length or an allowable degree of likeness can participate in the crossover. 6. Count preserving Crossover (CPC): This operator is based on the position of bits (0 or 1) that are different and offspring are generated based on the exchange of bits. 7. Elitist Crossover (EC): In this operator, at first, the population is shuffled randomly, then from each sequential pair of parent vectors, two new offspring are generated by crossover. 8. Self-Adaptive Simulated Binary Crossover (SBX): This operator updates the distribution index in the SBX crossover. 9. Scanning Crossover (scan), Uniform scanning crossover (U-Scan), Occurrence Based scanning Crossover (OB-Scan), Fitness Based Scanning Crossover (FB-Scan).

Also, some crossover operators have been proposed recently: The cross average crossover (CAX) with a rank-based selection method has been proposed[5]. A Gaussian crossover operator (GCO) for the generation of offspring using the Gaussian process has been proposed[6]. A new differential evolution crossover (DEC) has been proposed to avoid premature convergence in search space[7].

3 Proposed Algorithm

In this section, the proposed method is described as follows: The fundamental concepts are described in Section 3.1. The problem statement illustrates in Section 3.2. The framework of the proposed algorithm illustrates in Section 3.3.

3.1 Fundamental Concepts

The estimation of distribution algorithms (probabilistic model-building genetic algorithms) guides the search through creating and sampling an explicit probabilistic model of promising solutions. In inverse-modeling of multi-objective evolutionary algorithm (IN-MOEA), the distribution pattern (probabilistic model) of parents ((X^p) and (Y^p)) are acquired and are estimated by the conditional probability distribution of P($X^p|Y^p$) through the inverse modeling and then with this pattern, to generate offspring, some samples are produced in the objective space based on the distribution of parents Y^o. Then Y^o, using $P(X^p|Y^p)$ are transformed back to the decision space using the Bayes' theorem:

$$P\left(X^{o}\right)=\frac{P\left(X^{p} \mid Y^{p}\right) P\left(Y^{o}\right)}{P\left(Y^{p} \mid X^{p}\right)} \tag{4}$$

Where $P(X^p|Y^p)$, is a priori knowledge [8]. Estimation of m-input and n-output inverse-model $P(X^p|Y^p)$, will be time-consuming, where m and n are respectively the numbers of objectives and decision variables. This method applies a random grouping

strategy because all decision variables are independent of each other. Therefore, some decision variables are grouped to be derived from the same objective using inverse models. Inverse modeling can be approximated as follows:

$$P(X \mid Y) \approx \prod_{i=1}^{n} \left(P\left(x_i \mid f_j\right) + \varepsilon_{j,i}\right) \tag{5}$$

Where $j = 1, 2, \ldots, m, m > 2$, and it is assumed that $\varepsilon_{(j,i)} \sim N(0, (\sigma_n)^2)$, is Gaussian noise. So, the Gaussian process is used in inverse modeling:

$$P\left(x_i \mid f_j\right) = N\left(0, C + \left(\sigma_n\right)^2 I\right) \tag{6}$$

In linear regression, there is a linear relationship between the independent variables (our features) and the dependent variable (our target). When the error of the regression model has a normal distribution and is assumed a particular form of the prior distribution, then explicit results are available for the posterior probability distributions of the model's parameters. In fact, in linear regression, there is the mean of the conditional distribution of y_i given a $k \times 1$ predictor vector X_i as follows:

$$y_i = X_i^T W + \varepsilon_i \tag{7}$$

Where W is a prior and $\varepsilon_i \sim N(0, \sigma^2\ )$ and the likelihood function is equal to $P(y|X, W, \sigma^2)$ which is illustrated in formula 6.

3.2 Problem Statement

One type of crossover is simulated-binary-crossover (SBX). In this operator, the gene values of offspring have the same distance from the average gene values of parents and each point of the chromosome has the same probability to be selected as a crossover point. The offspring values a real-coded has been computed by Deb k. et, al. as follows [9]:

$$\begin{aligned} c_1 &= \bar{x} - \frac{1}{2}\beta\left(p_2 - p_1\right) \\ c_2 &= \bar{x} + \frac{1}{2}\beta\left(p_2 - p_1\right) \\ \bar{x} &= \frac{1}{2}\left(p_1 + p_2\right) \end{aligned} \tag{8}$$

The β is a spreading factor and was defined by Deb k. et, al. as the ratio of offspring spread points than parent points:

$$\beta = \left| \frac{off\ srping_1 - off\ spring\ _2}{\text{parent}\ _1 - \ \text{parent}\ _2} \right| \tag{9}$$

If the $\beta < 1$, the crossover is contracting, and if $\beta > 1$, the crossover is expanding, and if the $\beta = 1$, a crossover is stationary. Figure 2 shows the relation between β and parent and offspring:

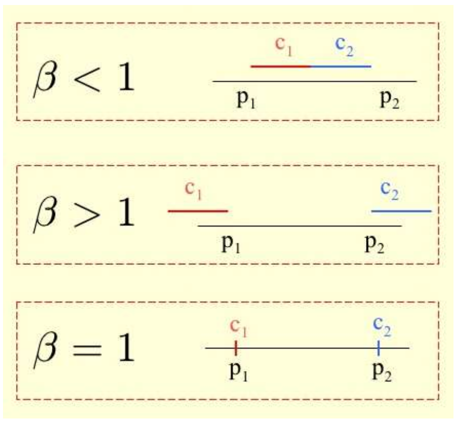

Figure 2. The Relation Between β and Parent and offspring.

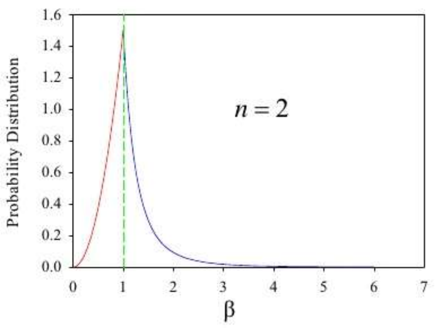

Figure 3. The Probability Distribution Function of $\beta, (p(\beta_i))$.

The simulated-binary-crossover (SBX) uses a probability distribution function [9]:

$$p\left(\beta_i\right) = \begin{cases} 0.5(n+1)\beta_i^n, & \text{if } \beta_i \le 1 \\ 0.5(n+1)\frac{1}{\beta_i^{n+2}}, & \text{otherwise} \end{cases} \tag{10}$$

Where n is the distribution index and it can be any real non-negative number. Figure 3 shows the probability distribution function of,$(p(\beta))$:

In Figure 3, the probability increases, when the β is increased towards one. Therefore, the probability of generating children closer to the parent is higher than that of points far away from the parents. Also, the distribution for $\beta \ge 1$, the probability of children's generation having a large β is small.

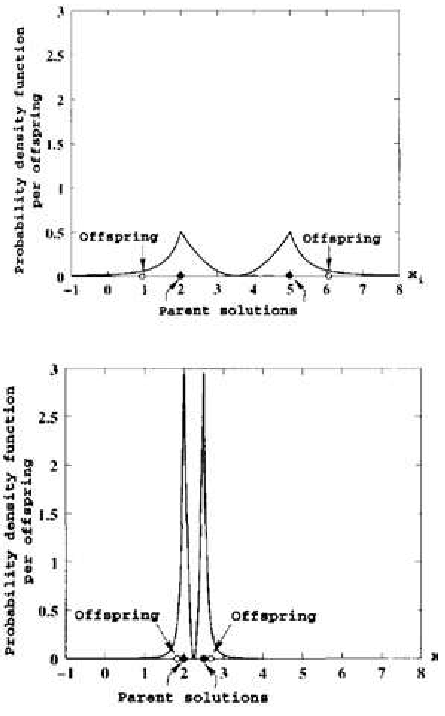

Figure 4. The small (a) and large (b) values of n and their effect on the distance of offspring from the parent.

The large values of n tend to create offspring which are closer to the parents and the small values of n allow the offspring to be far from parents, as illustrated in Figure 4.

To compute the value of β for generating the offspring, the simulated binary crossover uses a unified random number, $u \in [0,1)$, that way the area under the curve of probability from zero to β is equal to the value of u. Therefore β is computed as follows:

$$\beta = \begin{cases} (2u)^{\frac{1}{n+1}}, & \text{if } u \leq 0.5 \\ (\frac{1}{2(1-u)})^{\frac{1}{n+1}}, & \text{otherwise} \end{cases} \tag{11}$$

Figure 5 [9] shows the probability distributions of β for different values n.

The value of β has been computed from its probability distribution function through u which is a unified random number.

Therefore, the equation of (11) is the probability density function and the equation of (10) is the probability distribution function of β. In Figure 5 the probability distributions of β for different values n illustrate the dependence of probability distribution on n. Therefore, a distribution can be considered as a probability distribution function that covers all states of n in Figure 6.

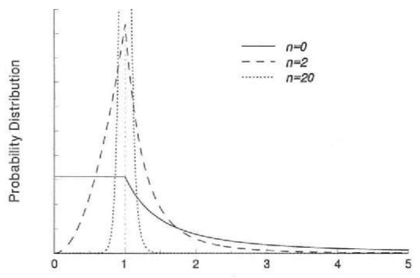

Figure 5. The Probability Distributions of β for Different Values n, $(n = 0, n = 2, n = 20)$.

3.3 Framework of the Proposed Algorithm

In simulated binary crossover (SBX) the offspring are generated from parents with a coefficient of variation. Offspring are generated by multiplying the β coefficient by parents. That is a linear relationship between parents and offspring: $x_i^{(t+1)} = W_i x_i^t$ which W_i and x_i^t are independent variables and $x_i^{(t+1)}$ is a dependent variable. Choosing the right coefficient of W_i can improve the diversity and convergence of offspring. But so far, linear regression has not been used to model a relationship between parents and offspring in crossover operations. In this paper, we concentrate on the Bayesian linear regression to generate offspring from parents in crossover operator, and a new approach is proposed: Gaussian Linear Regression crossover (GLR-C) which a Gaussian process is used for the normal linear regression model. The probability distributions of β are normal (Gaussian) distributions and the Gaussian probability density function is used for the probability distribution function of β and the probability distribution of the proposed method is on objective space in the mating pool. The Gaussian process is a technique for performing probabilistic regression. The framework of the proposed algorithm is illustrated in Figure 6.

The estimation of distribution algorithms (probabilistic model-building genetic algorithms) guides the search through creating and sampling an explicit probabilistic model of promising solutions. In inverse-modeling of multi-objective evolutionary algorithm (IN-MOEA), the distribution pattern (probabilistic model) of parents $((X^p) and (Y^p))$ are acquired and are estimated by the conditional probability distribution of $P(X^p|Y^p)$ through the inverse modeling and then with this pattern, to generate offspring, some samples are produced in the objective space based on the distribution of parents Y^o. Then Y^o, using $P(X^p|Y^p)$ are transformed back to the decision space using the Bayes' theorem:

$$P(X^o) = \frac{P(X^p \mid Y^p) P(Y^o)}{P(Y^p \mid X^p)} \tag{12}$$

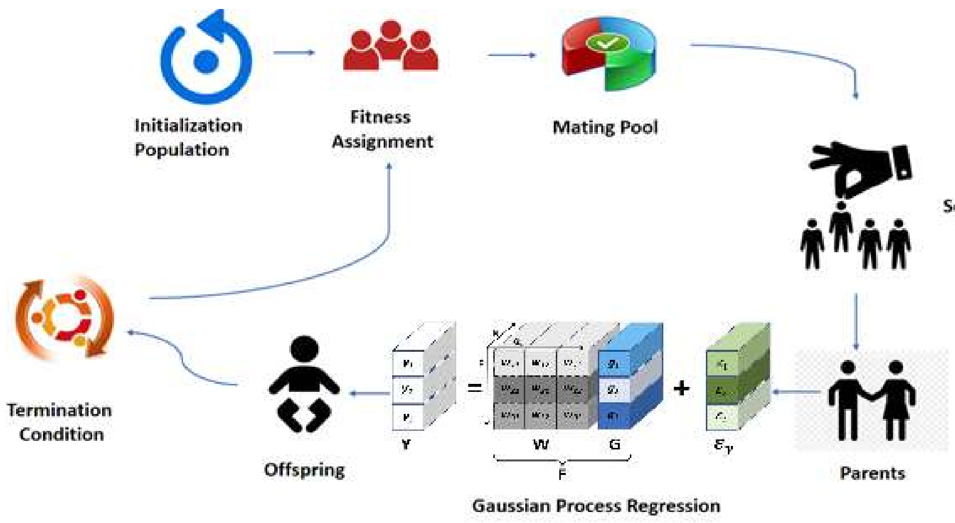

Figure 6. The Framework of the Proposed Algorithm.

Where $P(X^p|Y^p)$, is a priori knowledge [8]. Estimation of m-input and n-output inverse-model $P(X^p|Y^p)$, will be time-consuming, where m and n are respectively the numbers of objectives and decision variables. This method applies a random grouping strategy because all decision variables are independent of each other. Therefore, some decision variables are grouped to be derived from the same objective using inverse models. Inverse modeling can be approximated as follows:

$$P(X \mid Y) \approx \prod_{i=1}^{n}(P(x_i \mid f_i) + \varepsilon_{j,i}) \qquad (13)$$

Where $j = 1, 2, \ldots, m, m > 2$, and it is assumed that $\varepsilon_{j,i} \sim N(0,(\sigma_n)^2)$, is Gaussian noise. So, the Gaussian process is used in inverse modeling:

$$P(x_i \mid f_i) = N(0, C + (\sigma_n)^2 I) \qquad (14)$$

In linear regression, there is a linear relationship between the independent variables (our features) and the dependent variable (our target). When the error of the regression model has a normal distribution and is assumed a particular form of the prior distribution, then explicit results are available for the posterior probability distributions of the model's parameters. In fact, in linear regression, there is the mean of the conditional distribution of y_i given a $k \times 1$ predictor vector X_i as follows:

$$y_i = X_i^T W + \varepsilon_i \qquad (15)$$

Where W is a prior and $\varepsilon_i \sim N(0, \sigma^2)$ and the likelihood function is equal to $P(y|X, W, \sigma^2)$ which is illustrated in formula 14.

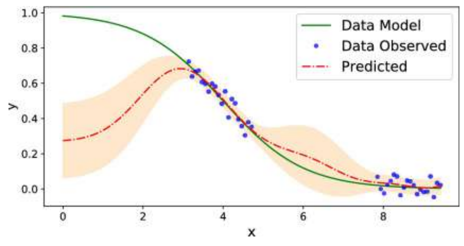

Figure 7. A Simple Example of the Proposed Method.

Therefore, at first, parents are selected from the mating pool. Then offspring is generated as follows:

$$\begin{cases} x_i^{(1,t+1)} - \varepsilon = 0.5[W_{i1}x_i^{(1,t)} + W_{i2}x_i^{2,t}] \\ x_i^{(2,t+1)} - \varepsilon = 0.5[W_{i2}x_i^{(1,t)} + W_{i1}x_i^{2,t}] \end{cases} \qquad (16)$$

The coefficient values of W_{i1} and W_{i2} can be obtained by calculating the value of each offspring.

In Figure 7, "Data Observed" is the parent, "Data Model" is the distribution function of parents, and the "Predicted" are offspring.

The overall algorithm framework of the proposed method (Gaussian Linear Regression crossover (GLR-C)), based on the Gaussian process regression model, is illustrated in Algorithm 1.

4 Experimental Analysis

In this section, to evaluate the proposed method, it is tested on benchmark tests suits and is compared with Gaussian Crossover Operator[6] (GCO) and Simulated Binary Crossover (SBX) on the Computational Expensive Optimization (CXO) benchmark tests[10] and the experimental results are analyzed by special

Algorithm 1 The Overall Algorithm Framework of the Proposed Method, Based on the Gaussian Process Regression Model.

```
1: Initialization: randomly initialize the population.
2: /*main loop*/
3: while the termination condition is not satisfied do
4:     Fitness Assignment: to determine the quality of solutions;
5:     Mating Pool: The process of selection of the best individuals based on their quality (fitness values) is done in the mating pool;
6:     Selection: the high-quality individuals have a higher probability of being selected in the mating pool.
7:     Parent: The parents in the mating pool are selected with each of the following methods: Elitist selection, Roulette wheel selection, Scaling selection, Tournament selection;
8:     Recombination: The selected parents from the mating pool are applied to generate offspring with the proposed method (Gaussian Linear Regression crossover (GLR-C)), and mutation methods which this process is called recombination; $P(X^{offspring}|X^{parrent}) \sim N(X^{offspring}|W^T X^{parrent}, \sigma^2)$
9:     Update the combined population: $P(t+1) = U_(k=1)^K (P^k(t) \cup O^k(t))$;
10:    t=t+1;
11: end while
```

performance metrics.

4.1 Benchmark Test Problems

The proposed method is tested on Computational Expensive Optimization (CXO) benchmark tests[10], which are used for the results of GCO. These test suites contain uni-modal, continuous, discrete, and separable or non-separable functions and are scalable for 10, 20, and 30 decision variables. The test problems have been illustrated in Table 1.

4.2 Performance Metrics

The Quality indicators (QIs) are used to evaluate the quality of solution sets and they are classified into six categories [11]: 1) QIs for convergence, 2) QIs for spread, 3) QIs for uniformity, 4) QIs for cardinality, 5) QIs for both spread and uniformity, and 6) QIs for incorporated quality of the four quality aspects.

In experiments, to evaluate the performance of algorithms, the Inverted Generational Distance (IGD) metric is considered which evaluates the convergence and the spread of the resulted PFs. It is tacked that F^* is a set of uniformly distributed solutions sampled from the PF, and P marks the approximated solutions created by a given algorithm, then the IGD value is estimated by:

$$f_i(x) = \frac{\sum_{f^* \in F^*} dist(f^*, P)}{|F^*|} \tag{17}$$

Where $|F^*|$ is the cardinality of F^* and dist(f^*,P) marks the shortest Euclidean distance from f^* to the elements in P. Also, the smaller the IGD value, the better is the quality of P.

$$\psi_k(f_k, f_{min}, f_{avg}) = \frac{2}{1 + e^{\frac{f_k - f_{min}}{-f_{avg}}}} - 0.5 \tag{18}$$

$$\Theta_i = min\left(\gamma\sigma_i, \frac{|S_i|}{6}\right) \tag{19}$$

4.3 Parameter Setting

In this work, QIs of IGD is applied to measure the performance on test instances F1 to F8, and 100 uniformly distributed points are selected from the PF of each test instance to be P^*.

The population size is adjusted to 100 for the proposed algorithm and SBX and GCO. We carried out 20 independent runs for each compared algorithm on each test instance. The final condition for each algorithm is adjusted to a maximum of 100000 fitness evaluations for all the test instances.

For all the competing algorithms, the same parameter setting of problems is applied. Parameter adjustments for these compared algorithms as summarized in Table 2. In this table, the different number of decision variables has been shown with D, to express the effect of different amounts of decision variables on each method. Also, in Simulated Binary Crossover (SBX), the effect of different distribution indexes has been compared.

The child's gene PDF is a result of the multiplication of its parent's distributions.

The Wilcoxon rank-sum test is adapted to compare the results obtained by our proposed algorithm and the other four algorithms at an important level of 0.05. In the tables that summarize the statistical results, the first and second lines illustrate the mean values and the standard deviations, respectively. As a result of the Wilcoxon rank-sum test, a + labeled means that the compared algorithm is better than the proposed algorithm; by contrast, a - labeled means that the proposed algorithm is better than the compared

Table 1. The Computational Expensive Optimization (CXO) Test Problems.

Name	Function	Range	Properties
Sphere	$F_1 = \sum_{i=1}^{D} x_i^2$	[-5.12,5.12]	Unimodal
Ellipsoid	$F_2 = \sum_{i=1}^{D} ix_i^2$	[-5.12,5.12]	Unimodal
Rotated Ellipsoid	$F_3 = \sum_{i=1}^{D} \sum_{j=1}^{i} x_j^2$	[-5.12,5.12]	Unimodal
Step	$F_4 = \sum_{i=1}^{D} ([x_i + 0.5])^2$	[-5.12,5.12]	Unimodal, Discontinuous
Ackley's	$F_5 = -20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)\right) + 20 + e$	[-32,32]	Multi-modal
Griewank's	$F_6 = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} cos(\frac{x_i}{\sqrt{i}}) + 1$	[-600,600]	Multi-modal
Rosenbrock's	$F_7 = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$	[-2.048,2.048]	Multi-modal, Non-separable, Having a very narrow valley from local optimum to global optimum
Rastrigin's	$F_8 = \sum_{i=1}^{D} (x_i^2 - 10cos(2\pi x_i) + 10)$	[-5.12,5.12]	Multi-modal

Table 2. Parameter Settings of the Four Algorithms in Comparison.

Algorithm	Gaussian Linear Regression crossover (GLR-C)	Simulated Binary Crossover (SBX)	Gaussian Crossover Operator (GCO)
Parameter settings	D=10,20,30	The distribution index, n=2,10,20 D=10,20,30	D=10,20,30

algorithm; while a ≈ labeled means that there is no statistically significant difference between the result obtained by the proposed algorithm and the compared algorithm.

The IGD indicator in Table 3 compares the statistical results of QIs for investigating convergence, spread, cardinality, and uniformity on test benchmarks. In addition, Figures 7, 8, and 9 show the resulting IGD curves of F1 to F4, for different values of parameters, in which the horizontal and vertical axes are test instances and IGD values, respectively.

From the content of Table 3, it can be illustrated that the proposed algorithm (e.g, Gaussian Linear Regression crossover (GLR-C)) on test functions is better than Simulated Binary Crossover (SBX) and Gaussian crossover (GCO) based on IGD performance metric. By looking at the diagrams in Figures 8, 9, and 10, it can be seen that the Gaussian Linear Regression crossover (GLR-C) is better than Simulated Binary Crossover (SBX) and Gaussian Crossover Operator (GCO) in 6 cases in IGD performance metric. Also, by increasing the dimensional of individuals in the population, this result remains unchanged. In compared algorithms, the different values of n, in Simulated Binary Crossover (SBX), are considered. So, the effect of different values of this parameter is shown in our evaluation. By looking at the diagrams in Figures 11, 12, and 13, it can be seen that the distribution of the Gaussian Linear Regression crossover (GLR-C) is better than Simulated Binary Crossover (SBX) in different values of n.

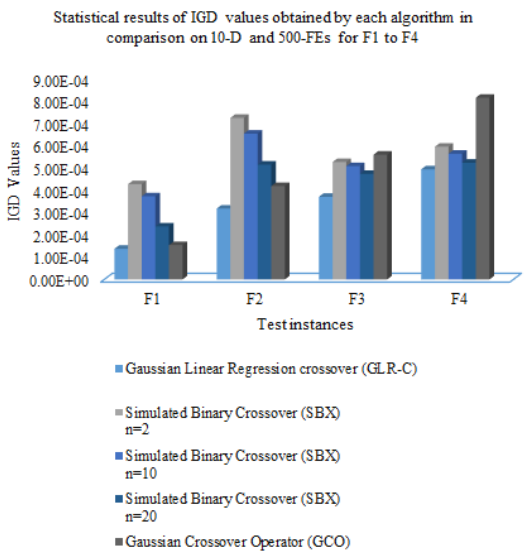

Figure 8. Statistical Results of IGD Values Obtained by Each Algorithm in Comparison on 10-D F1 to F4.

Therefore, the convergence, spread, cardinality, and uniformity of the proposed algorithm are better than the two compared algorithms.

Table 3. Statistical Results of IGD Values Obtained by Each Algorithm in Comparison to the Test Problem.

Problem	D	Fes	Gaussian Linear Regression crossover (GLR-C)	Simulated Binary Crossover (SBX)			Gaussian Crossover Operator (GCO)
				n=2	n=10	n=20	
F_1	10	500	1.3688e-4 (1.63e-2)+	4.2591e-4 (1.41e-2)-	3.7143e-4 (3.81e-2)-	2.3669e-4 (2.18e-2)-	1.5359e-4 (2.71e-2)
	20	1000	3.7040e-4 (1.58e-2)+	7.1721e-4 (3.03e-2)+	8.6491e-4 (1.68e-2)-	5.9008e-4 (3.81e-2)-	4.8372e-4 (2.24e-2)
	30	1500	4.9799e-4 (1.01e-2)+	8.4628e-4 (3.02e-1)-	6.9218e-4 (1.78e-1)-	6.0237e-4 (2.12e-1)-	5.4329e-4 (1.29e-1)
F_2	10	500	3.1659e-4 (1.63e-2)+	7.2148e-4 (2.13e-2)-	6.5112e-4 (3.19e-2)-	5.1284e-4 (2.41e-2)-	4.1734e-4 (1.81e-2)
	20	1000	3.9141e-3 (1.63e-2)+	4.8531e-3 (1.13e-2)-	4.7121e-3 (3.31e-2)-	4.3567e-3 (2.66e-2)-	4.0361e-3 (1.63e-2)
	30	1500	5.2082e-3 (3.21e-2)+	8.2082e-3 (2.14e-2)-	7.7126e-3 (1.95e-2)-	7.6154e-3 (3.01e-1)-	6.6154e-3 (4.02e-1)
F_3	10	500	3.6913e-4 (1.57e-2)+	5.2456e-4 (1.33e-2)-	5.0612e-4 (3.19e-2)-	4.7168e-4 (2.36e-2)+	5.5716e-4 (1.63e-2)
	20	1000	1.8184e-3 (2.93e-2)-	5.5491e-3 (1.83e-2)-	4.3761e-3 (2.56e-2)-	3.4917e-3 (2.04e-2)+	6.5012e-3 (2.65e-2)
	30	1500	3.8638e-3 (1.71e-2) +	8.2082e-3 (2.14e-2)-	7.7126e-3 (1.95e-2)-	7.2631e-3 (5.02e-1)+	7.2631e-3 (2.09e-1)
F_4	10	500	4.9176e-4 (3.02e-2)+	5.9274e-4 (2.71e-2)-	5.6139e-4 (1.86e-2)-	5.2145e-4 (2.54e-2)+	8.1134e-4 (2.61e-2)
	20	1000	3.5619e-3 (3.01e-2)+	3.8912e-3 (2.76e-2)-	3.6875e-3 (2.17e-2)-	3.5981e-3 (3.34e-2)+	7.5948e-3 (1.33e-2)
	30	1500	3.2736e-3 (1.71e-2)+	7.6784e-3 (1.87e-2)-	6.9136e-3 (2.25e-2)-	6.3418e-3 (2.81e-1)-	8.4174e-3 (2.17e-1)
F_5	10	500	6.5145e-5 (3.11e-2)+	9.1237e-5 (1.76e-2)-	8.5438e-5 (1.57e-2)-	7.8711e-5 (1.91e-2)-	6.9812e-5 (2.27e-2)
	20	1000	8.2371e-5 (2.65e-2)+	8.9178e-5 (2.81e-2)-	8.7659e-5 (2.44e-2)-	8.638e-5 (2.91e-2)-	8.4141e-5 (2.38e-2)
	30	1500	7.6169-5 (1.38e-2)+	9.4561e-5 (2.84e-2)-	9.2378e-5 (1.08e-2)-	9.081e-5 (3.22e-1)-	7.9821e-5 (1.48e-1)
F_6	10	500	4.4912e+1 (4.02e+0)-	2.9163e+1 (2.33e+0)-	2.8561e+1 (3.29e+2)-	2.7823e+1 (1.46e+0+	1.0570e+1(1.63e-2)
	20	1000	5.5081e+1 (3.04e+0)-	4.8531e+1 (3.61e+0)-	3.7428e-3 (2.31e+0)-	3.2456e+1 (3.68e+0)	1.2075e+1(4.02e+0)
	30	1500	4.9347e+1 (2.87e+0)-	5.9167e+1 (2.24e+0)-	5.6549e+1 (2.65e+0)-	5.1278e+1 (3.64e+0)-	1.7541e+1(2.71e+0)
F_7	10	500	3.0495e+2 (3.09e+0)-	2.9464e+2 (4.71e+0)-	2.7617e+2 (3.29e+0)-	2.3481e+2 (4.32e+0)-	1.7661e+2(2.35e+0)
	20	1000	6.3820e+2 (4.03e+0)-	5.6711e+2 (2.21e+0)-	4.9713e+2 (3.19e+0)-	4.1951e+2 (2.28e+0)-	1.8642e+2(3.84e+0
	30	1500	6.8719e+2 (2.78e+1)-	5.6561e+1 (2.24e+0)-	5.5690e+1 (1.95e+0)-	5.3284e+2 (2.06e+0)-	2.1828e+2(3.33e+0)
F_8	10	500	5.1182e-2 (3.21e-2)-	6.5471e-2 (1.43e-2)-	5.9648e-2 (3.27e-2)-	5.4179e-2 (2.93e-2)-	5.3209e-2 (1.91e-2)
	20	1000	4.8906e-2 (2.11e-2)-	4.5618e+1 (3.52e-2)-	4.3198e-3 (2.31e-2)-	4.0121e-2 (2.21e-2)-	3.8182e-2 (1.87e-2)
	30	1500	2.5351e-1 (1.63e-2)-	5.9637e-1 (2.24e+0)-	5.6913e-1 (2.65e+0)-	5.3480e-1 (1.73e+0) -	3.0180e-1(4.13e+0)
+/-/=			5/3/0	0/8/0	0/8/0	3/5/0	

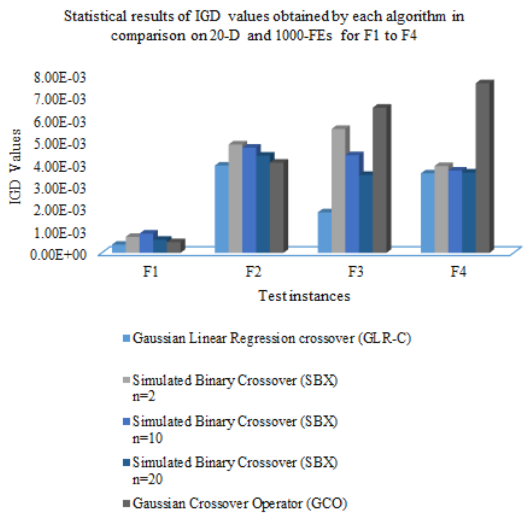

Figure 9. Statistical Results of IGD Values Obtained by Each Algorithm in Comparison on 20-D F1 to F4.

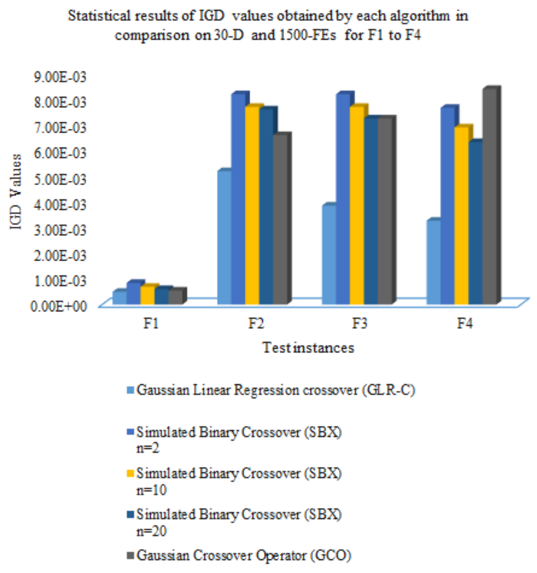

Figure 10. Statistical Results of IGD Values Obtained by Each Algorithm in Comparison on 30-D F1 to F4.

Figure 13 illustrates that in Simulated Binary Crossover the runtime is growth high, while in the proposed method, the growth of runtime is uniformly almost. Although, the traditional crossover (such as SBX) have low time consuming than the Gaussian crossover (GCO) and the proposed method (Gaussian Linear Regression crossover (GLR-C) when the number of decision variables is low, but with an increase in the number of decision variables is observed that the run time of GCO and the proposed method (GLR-C) is better than the other comparison algorithms.

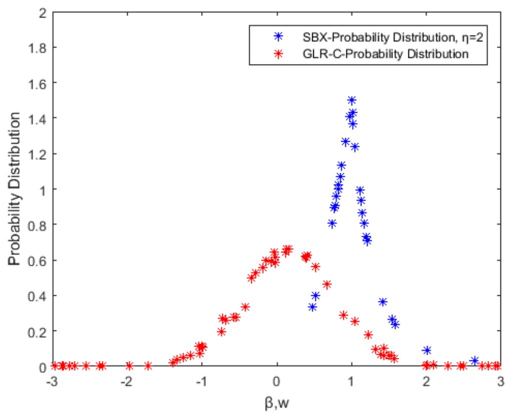

Figure 11. Probability Distribution of SBX Operator for η=2 and GLR-C on 30-D for F2.

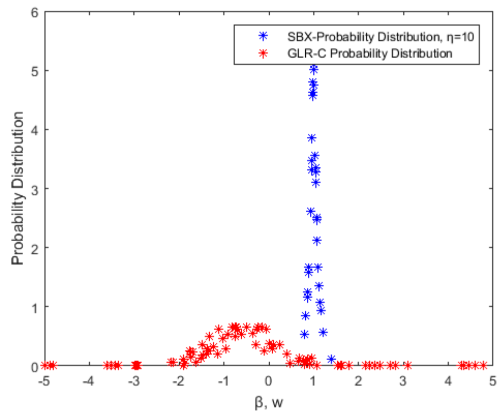

Figure 12. Probability Distribution of SBX Operator for η=10 and GLR-C on 30-D for F2.

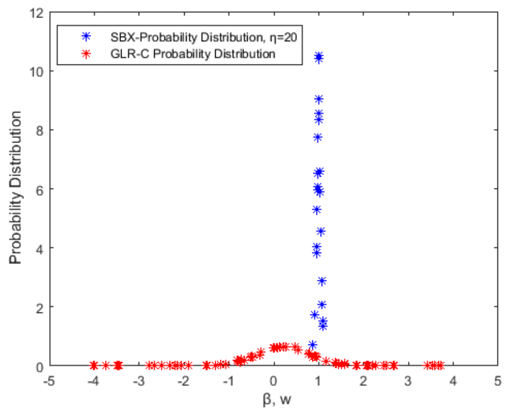

Figure 13. Probability Distribution of SBX Operator for η=20 and GLR-C on 30-D for F2.

5 Conclusions

In this paper, a Gaussian linear regression crossover (GLR-C) has been proposed. In simulated binary crossover (SBX) the offspring are generated from parents with a coefficient of variation. Offspring are gener-

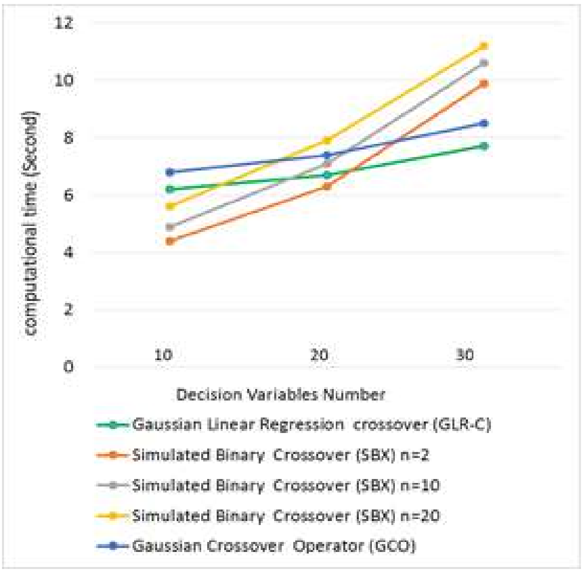

Figure 14. The Averaged Over 20 Run Times of the Proposed and Compared Methods Run on F4.

ated by multiplying the β coefficient by parents. The simulated-binary-crossover (SBX) uses a probability distribution function for the β coefficient. There is a linear relationship between parents and offspring: $x_i^{((t+1))} = W_i x_i^t$ which W_i and x_i^t are independent variables and $x_i^{((t+1))}$ is a dependent variable. Choosing the right coefficient of W_i can improve the diversity and convergence of offspring. The idea is to apply linear regression to model a relationship between parents and offspring in crossover operations through the Gaussian process. The reason for using this process is that the probability distribution of the SBX operator is based on the parent in the mating pool on decision space, while the probability distribution of the proposed method is on objective space in the mating pool. Therefore, the distribution of the proposed method is better than the SBX operator. In addition, the spread, uniformity, and convergence of distribution in the GLR-C are better than the SBX operator. To optimize problems on the combinatorial sets, the proposed method is applied. The performance of the proposed algorithm was tested on Computational Expensive Optimization (CXO) benchmark tests. The proposed method performs robustly competitively on a variety of test instances compared to two representative operators. The best assignment of x_n to f_m instead of random grouping, constructing the Gaussian process in an inverse model can be applied in future works. Also, this method can replace to solve genetic optimization problems. Some of these are as follows: trust inference in social networks by the combination of neural network and genetic algorithm[12], novel diversity-preservative strategies for genetic algorithms and their application for large-scale optimization[13], multi-objective genetic algorithm-based ensemble classifier using classification error, sparsity, diversity and density criterion[14].

References

[1] V. K. Patel, V. J. Savsani, and M. A. Tawhid. Metaheuristic methods. In *Thermal System Optimization*, pages 7–32. Springer, 2019.

[2] M. Rostami, K. Berahmand, and S. Forouzandeh. A novel community detection based genetic algorithm for feature selection. *IEEE Access*, 8(1): 1–27, 2021. doi:https://doi.org/10.1186/s40537-020-00398-3.

[3] M. Rostami, K. Berahmand, and S. Forouzandeh. Crossover and Mutation Operators of Genetic Algorithms. *International Journal of Machine Learning and Computing*, 7(1):9–12, 2017. doi:10.18178/ijmlc.2017.7.1.611.

[4] A. J. Umbarkar and P. D. Sheth. CROSSOVER OPERATORS IN GENETIC ALGORITHMS: A REVIEW. *ICTACT Journal on Soft Computing*, 6(1), 2015. doi:10.18178/ijmlc.2017.7.1.611.

[5] M. Y. Orong, A. M. Sison, and R. P. Medina. A new crossover mechanism for genetic algorithm with rank-based selection method. In *2018 5th International Conference on Business and Industrial Research (ICBIR)*, pages 83–88. IEEE, 2018. ISBN 978-1-5386-5253-4. doi:10.1109/ICBIR.2018.8391171.

[6] M. Sain, Y. J. Kang, and H. J. Lee. An introduction to a novel crossover operator for real-value encoded genetic algorithm: Gaussian crossover operator. In *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, pages 85–90. IEEE, 2018. ISBN 978-1-5386-6143-7. doi:10.1109/IIPHDW.2018.8388331.

[7] M. Z. Ali, N. H. Awad, P. N. Suganthan, A. M. Shatnawi, and R. G. Reynolds. An improved class of real-coded Genetic Algorithms for numerical optimization. *Neurocomputing*, 275:155–166, 2018. doi:10.1016/j.neucom.2017.05.054.

[8] R. Cheng, Y. Jin, K. Narukawa, and B.Sendhoff. A Multiobjective Evolutionary Algorithm Using Gaussian Process-Based Inverse Modeling. *IEEE Transactions on Evolutionary Computation*, 19(6):838 – 856, 2015. ISSN 1941-0026. doi:10.1109/TEVC.2015.2395073.

[9] K. Deb and R. B. Agrawal. Simulated Binary Crossover for Continuous Search Space. *Complex Systems*, 9(3):115–148, 2000.

[10] Y. Ding, Y.Wu, C. Huang, S. Tang, Y. Yang, L. Wei, Y. Zhuang, and Q. Tian. Learning To Learn by Jointly Optimizing Neural Architecture and Weights. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern*

Recognition, pages 129–138, 2022.

[11] M. Li and X. Yao. Quality Evaluation of Solution Sets in Multiobjective Optimisation: A Survey. *ACM Computing Surveys (CSUR)*, 52(2):1–38, 2019. doi:10.1145/3300148.

[12] M. Fayyaz, H. Vahdat-nejad, and M. Kherad. Trust Inference in Social Networks by Combination of Neural Network and Genetic Algorithm. *Tabriz Journal of Electrical Engineering*, 50(1): 331–340, 2020. doi:10.1145/3300148.

[13] H. Ismkhan. Novel Diversity-Preservative Strategies for Genetic Algorithms and Its Application for Large-Scale Optimization. *TABRIZ JOURNAL OF ELECTRICAL ENGINEERING*, 48(2): 467–479, 2018.

[14] B. Zamani Dehkordi and Z. Nekouei. Multi Objective Genetic Algorithm Based Ensemble Classifier Using Classification Error, Sparsity, Diversity and Density Criterion. *TABRIZ JOURNAL OF ELECTRICAL ENGINEERING*, 47(4):1479–1487, 2018.

Pezhman Gholamnezhad received his Ph.D. in the field of artificial intelligence computer engineering from Islamic Azad University, South Tehran Branch in 1399. His areas of interest are evolutionary processing, machine learning, statistical pattern recognition, and fuzzy systems.