IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 2 March 2023, accepted 28 April 2023, date of publication 2 May 2023, date of current version 9 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3272356

RESEARCH ARTICLE

A Stage-Specific Gene Expression Framework for Promoting Genetic Algorithm

PAN FENG AND HAOZHOU SU
College of Data Science and Information Engineering, Guizhou Minzu University, Guiyang, Guizhou 550025, China
Key Laboratory of Pattern Recognition and Intelligent System, Guizhou Minzu University, Guiyang, Guizhou 550025, China

Corresponding author: Haozhou Su (haozhou.su@gmail.com)

This work was supported by the Teaching Reform Fund Project of Guizhou Province of China under Grant 2022047 and Grant 2022015.

ABSTRACT Genetic algorithm (GA) has become very popular for tackling computationally expensive numerical optimization problems. To optimize their performance and convergence rate, we propose and demonstrate an embedded stage-specific gene expression framework (SGEF) with a two-tier coding scheme and a filter operator based on variational auto-encoder. For the first time, the structure of variational auto-encoder is used to map the parent population of a genetic algorithm from the uniformly distributed high dimensions to the Gaussian distributed low dimensions. The design of the filter operator which can maintain the trade-off balance between exploitation and exploration is also analyzed in detail. The experimental results demonstrated the good performance of the proposed SGEF in promoting both single-objective optimization genetic algorithms and multiple-objective optimization genetic algorithms.

INDEX TERMS Genetic algorithm, variational auto-encoder, neural networks, stage-specific gene expression theory, exploration and exploitation.

I. INTRODUCTION

Over the past few decades, metaheuristic algorithms are used to solve the optimization problems in an effective manner. To name a few domains, polymer design in materials [1], detection of Parkinson disease in Clinical Medicine [2], mechanical design problems in engineering [3] and Transportation energy demand forecasting in Operations Research [4]. Among the metaheuristic algorithms, genetic algorithm (GA) is an effective algorithm which is inspired from Darwinian evolutionary theory. Genetic algorithm was proposed by John Holland in the 1960s. GA has been used to solve various NP-hard problems due to the advantages of self-optimizing, self-adaptive and self-learning. For example, GA shows the superior performance in various fields of multimedia, such as image processing, video gaming, and video processing. GA has been used to decomposing an image due to their search capability [5]. Katoch et al. utilized GA as image restoration technique [6]. In video processing, GA hybridized with neural network (NN) and support vector machine (SVM) can be used to resolve video segmentation problems which means distinguishing objects from the background. In video gaming, Shivgan and Dong formulate the Unmanned Aerial Vehicles path planning problem as a traveling salesman problem in order to optimize this problem with GA [7]. GA also has been applied to solve operation management (OM) problems [7], [8], [9]. Some of these well-known OM fields are scheduling, inventory control and facility layout problem. Abualigah and Alkhrabsheh proposed a multi-verse optimizer genetic algorithm (MVO-GA), which could significantly optimize the tasks' transfer time of a virtual machines-based central cloud facilities [8]. Huo et al. developed a nondominated sorting genetic algorithm with an adaptive local search operator to search for Pareto optimal solution of a multi-floor hospital facility layout problem [9]. Existing studies have shown that the modified genetic algorithms are effective in solving various NP-hard problems, and the metaheuristic strategy can generate acceptable solutions in the search space. But the accuracy of these algorithms will decrease and the speed will become slower when facing problems with muti-decision variables, which known as ''curse of dimension in high-dimensional data''.

The associate editor coordinating the review of this manuscript and approving it for publication was Qichun Zhang.

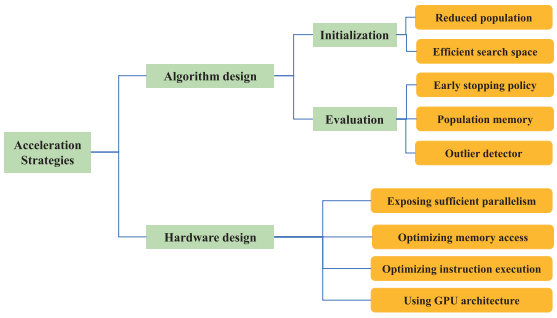


FIGURE 1. Different acceleration strategies.

There are two major challenges faced by GA. The first challenge is exploitation. GA is much more likely to generate ineffective chromosomes when generating populations without using prior experience. The second challenge is exploration, the process of generating offspring population from parent population is stochastic due to the operation flows of crossover operator, variation operator and selection operator. Exploitation and exploration are the key elements of GA. Trade-off balance between these two elements is always critical for GA to maintain ease of implementation, simplicity and ability to avoid local optima. Thus, this paper proposes a stage-specific gene expression framework (SGEF) aiming at optimizing the speed, efficiency and quality of the existing GA along with reducing the iteration times.

In our SGEF, a two-tier coding scheme and a population filter operator are combined to help existing GA achieve superior approximation performance. In addition, a new strategy for generating individual is presented based on variational auto-encoder, which can generate individuals from latent space. To summarize, the main contributions of this work are summarized as follows:

1) A new stage-specific gene expression framework (SGEF) is proposed for optimizing the speed, efficiency and quality of the genetic algorithms. SGEF introduces a two-tier coding scheme in which the original variables are considered as DNA and the compressed data are considered as massage RNA (mRNA). A strategy for generating individual based on this two-tier coding scheme named iterative variational auto-encoder (IVAE) is introduced to mimic the transcribing process in cells, which means copying a gene's DNA sequence to make an RNA molecule. This strategy can generate high-quality individuals using prior information learned from the parent population.
2) A population filter is proposed to simulate the selective degradation of mRNA in cells. In each generation, the individuals in parent population will be mapped into to a latent space by a iterative variational auto-encoder. The filter operator decides whether to filter the individual by measuring its fitness value and its distribution in the latent space.
3) We can embed SGEF into any genetic algorithm. To verify the effectiveness of SGEF, we embed SGEF into six well-known genetic algorithms and use seventeen constrained optimization problems as benchmark. The experiments show that the performance of these embedded algorithms is significantly improved compared to original algorithms.

The rest of this paper is organized as follows. Section II introduces the motivations and the related techniques of this paper. Section III details the proposed stage-specific gene expression framework. Section IV presents the experimental results of the SGEF with different genetic algorithm. Finally, section V gives the conclusions.

II. RELATED WORK AND MOTIVATIONS

To facilitate the understanding of the proposed stage-specific gene expression framework (SGEF), we first introduce a general genetic algorithm and some acceleration strategies, and then present some preliminaries of stage-specific gene expression theory and mRNA degradation phenomena. Finally, make a review of related work and give an motivation for this paper.

A. GENETIC ALGORITHMS AND ACCELERATION STRATEGIES

This subsection will give a brief overview on genetic algorithm (GA) and mainly focus on their acceleration strategies. GA is an algorithm inspired by the process of biological evolution and inheritance in nature. It is suitable for solving complex optimization problems by treating the solutions to the problem as a population of chromosomes (gens or individuals). A genetic algorithm is an iterative process that generally starts with a population of randomized individuals. The iterative process of genetic algorithm is called generation. For each generation, the fitness of every individual in the parent population is measured according to the objective function. A child population will be generated by crossover operators, mutaiton operators, and selection operators based on the previous population. This genetic algorithm ends when either a prescribed maximum number of generations has been reached or a satisfactory solution is found.

Genetic algorithms have been successfully applied to tackle problems involving a small number of decision variables. However, real-life optimization problems often have a large number of decision variables or have more than one objective function to be optimiazed simultaneously, which are known as a "Large-Scale Global Optimization" (LSGO) problem and multi-objective optimization. Genetic algorithms need a highly time-consuming evaluation to find the optimal solution in the large search space of these problems. In other words, This is a high computational overhead task for genetic algorithms to solve these problems. To overcome these challenges, many genetic algorithms with acceleration strategies have been proposed to speed up the optimization. The summarized acceleration strategies are shown in Fig 1.

From the perspective of algorithm design, we summarized current acceleration strategies in initialization and evaluation phases. In initialization phase, dividing the whole population into several sub-population and designing a efficient search apace can speed up computaion time significantly. Gu et al. proposed a hybrid genetic grey wolf algorithm (HGGWA) which embeds three genetic operators [10]. HGGWA can divide the whole population into several sub-populations and initialize them use an opposition-based learning strategy. Jia et al. proposed a two-stage genetic algorithm with local search (TSOL) [11]. In the first stage, TSOL can find a promising region in the whole search space. In the second stage, TSOL obtains the global optimal solution by searching thoroughly in the promising region. Wu et al. proposed a new genetic algorithm which can divide the initial large-scale problem into several small-scale subproblems, then use a hybrid search strategy to solve the subproblem respectively [12]. Minh et al. proposed a strategy for optimizing physical quantities based on exploring the latent space of a variational autoencoder (VAE) [13]. This algorithm trains a VAE using an existing dataset whose elements have good performance for fitness functions. These algorithms solve complex optimization problems by decomposing the search space before population iteration. In evaluation phase, researchers often use early stopping policy, population memory, and outlier detector to accelerate the GA. Ahmed et al. proposed methodology which can work on limited computing assets and decrease the overhead on the computing resources [14]. Sun et al. designed a cache component which can store elite individuals in each population during the optimization. to significantly accelerate the fitness evaluation given a limited computational resource [15]. Li et al. proposed a genetic algorithm based on variational auto-encoder (VAGA) to detect outliers in high-dimensional data [16]. VAGA constructs a variational auto-encoder (VAE) to reduce dimension in the first step, then searches the latent space of the trained VAE using a genetic algorithm for detecting outliers of the VAE layer.

These acceleration strategies in the perspective of algorithm design improve the search efficiency at the expense of sub-optimiality. For instance, dividing the initial large-scale problem into several small-scale subproblems sacrifice the ability to find the global optimal solution. Narrowing the search space may lead to data loss. According to the no free lunch theory, highly accurate algorithms are always time-consuming. To accelerating convergence speed of GAs without sacrificing accuracy, we can only deploy these algorithms on a more advanced and powerful hardware plateform.

From the perspective of hardware implementaion, we concluded that there are four acceleration strategies, including exposing sufficient parallelism, optimizing memory access, optimizing instruction execution and using GPU architecture. There are three basic approaches to expose parallelism, including master-slave model, island model, and cellular model [17]. Liu et al. proposed a dynamic immigration scheme and an interleaving emigration scheme which can reduce the communication overhead of migration [18]. Cheng and Gen discuss how to build up an efficient implementation of accelerating GAs on GPU-CPU heterogeneous architectures for hyper-scale computing [19]. These hardware platforms proposed above can significantly speed up the genetic algorithms' search processes under a reasonable utilization of computing resources.

B. STAGE-SPECIFIC GENE EXPRESSION THEORY

The existing genetic algorithms and their improvement are inspired by the macroscopic species competition process. If we observe gene expression from a microscopic perspective at the cellular level, genetic algorithms will be reconstructed architecturally. In the contemporary view of gene expression, the process of gene expression is a complex system, which includes two subsystems: transcription and translation. In transcription process, an RNA will be synthesized by information from a DNA. In the translation process, this RNA can produce final products which can ultimately affect a phenotype. These final products are proteins or non-coding RNA. Transcription and translation are functionally and physically connected to each other. This collaboration ensures that the massage can efficiently transfer from DNA to proteins with no individual step being omitted [20].

In the process of gene expression, the message RNA (mRNA) plays a role in connecting the transcription and the translation. mRNA is a type of RNA that can copy a segment of DNA and then use these messages to encode proteins. Counterintuitively, although mRNA is crucial in gene expression, the proportion of DNA sequences in the genome which can be transcribed into mRNA and then translated into proteins is very small [21]. Researchers found that about 50% of the genes in the human genome are duplicates, and only 2% are expressed as proteins.

C. MASSAGE RNA DEGRADATION PHENOMENON

The rapid changes in the pattern of gene expression depend on the amount of corresponding message RNA which is available for translation. The number of massage RNA is affected by the mRNA synthesis rate and degradation rate. Once the corresponding mRNA of a protein is degraded, the protein cannot be synthesized. The rate of mRNA synthesis depends on the transcription step which is explained in the previous subsection. In this subsection we introduce several unspecific mechanisms of degradation which can affect the decay of all mRNA species.

The mRNA degradation is a mechanism to eliminate mRNA. This mechanism works when mRNA either has aberrant features or is no longer required in its cells. In the past few years, different mechanisms have been proposed to explain the mRNA degradation phenomenon. Deutscher established the initial model by observing the changes of mRNA concentration in bacteria [22]. Apirion logically

proposed a model based on the initiation and coalescence of endonucleolytic [23]. Yokota claimed three subcategories of enzymes named CCR4-NOT, 5' exonucleases, 3' exonucleases can mediate mRNA degradation [24]. Deneke et al. proposed a general theoretical massage RNA expression model which focuses on the distributions of general mRNA lifetime [25].

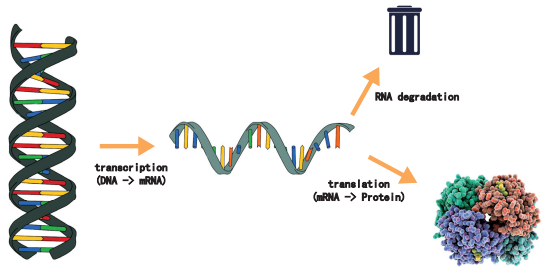


FIGURE 2. Information flow of gene expression.

D. REVIEW IN THIS SECTION

In this chapter, we first review the development of genetic algorithms and list some acceleration strategies. GA is an algorithm inspired by the process of biological evolution and inheritance in nature. In genetic algorithm, the mutation operator is inspired by genetic variations at the DNA level and the crossover operator is inspired by the combination of gametes in the fertilization process.

Next, we elaborate on gene expression process, which can enlighten us to improve the genetic algorithm framework. If we think of the process of gene expression as an information flow (fig 2), the information in DNA can flow into mRNA by transcription. This information may be lost as a result of RNA degradation, or it may flow into protein by translation [26]. During the expression of human genes, less than 2% of all DNA sequences can be expressed, which means that during the transcription, the information contained in the DNA is compressed and only the most valuable information can flow to the mRNA. By adding mRNA coding layers to genetic algorithm, we can use this theory to reduce the dimension of problem variables. Cells use mRNA degradation as one of the methods to regulate which genes will be expressed. The information contained in the mRNA will be lose when the mRNA either has aberrant features or is no longer required in its cells. At each iteration of the genetic algorithm, we can imitate this phenomenon to filter out useless individuals.

III. THE DETAILS OF SGEF

The proposed stage-specific gene expression framework (SGEF) tries to modify the existing genetic algorithm framework by mimicking gene expression in cells. It captures the related stage-specific gene expression theory and mRNA degradation phenomena which are described in section II. According to stage-specific gene expression theory, less than 2% of human DNA can be transcribed into mRNA. By adding mRNA coding layers to genetic algorithm, we can use this theory to reduce the dimension of problem variables. The translation and degradation of mRNA play an important role in controlling gene expression. In each generation of a genetic algorithm, we can design an improved select operator called filter operator, which can filter out abnormal individuals, imitate the mRNA degradation, and select potential individuals in the population. Next, the implementation of these theories and phenomena in terms of an accelerating framework is explained.

A. TWO-TIER CODING SCHEME

When facing high-dimensional optimization problems, a number of genetic algorithms first train an encoder and a decoder based on the initial population. This encoder is a dimensionality reduction method which can reduce the dimensionality of the high-dimensional variables by mapping the search space to a latent space. These genetic algorithms then search the latent space. The optimal solution obtained by these algorithms in the latent space will be mapped to the original space by the trained decoder. However, due to the different distribution of individuals in the initial population and the evolved population, the decoder and encoder trained by the initial population are no longer suitable for the new population.

We proposed a two-tier coding scheme. In this scheme, the original variables are treated as DNA using binary encoding scheme, and the data after dimensionality reduction is treated as massage RNA (mRNA) using value encoding scheme. To connect these two encoding tiers and implement this coding scheme, we proposed an iterative variational autoencoder (IVAE), which embeds variational autoencoders in each generation of a genetic algorithm. A VAE is a probabilistic autoencoder which takes high-dimensional input data and compresses it into a smaller latent space. The points in this latent space have a probability distribution, which are the mean and variance of a Gaussian in general. This model can map search space to a continuous, structured latent space, which is useful for generating optimal solutions in genetic algorithms. Data in latent space can be reconstructed by the decoder of VAE. In each iteration process of the SGEF, firstly, we use the well-adapted DNA codes in parent population training IVAE. Secondly, we compress all high-dimensional DNA codes from the parent population into mRNA code in latent space by the encoder of IVAE. Thirdly, we use traditional selection operator, crossover operator, mutation operator, and generation operator to manipulate these mRNA codes in latent space. Finally, the manipulated mRNA codes will be decoded by decoder to generate offspring populations. Algorithm 1 summarized the general procedure of a Two-tier coding scheme.

1) CONNECTIONS BETWEEN DNA AND mRNA

In two-tier coding scheme, the original variables are treated as DNA using binary encoding scheme, and the data after dimensionality reduction is treated as massage RNA (mRNA) using value encoding scheme. we use a variational autoencoder (VAE) to connect DNA codes and mRNA codes in

Algorithm 1 Pseudo-Code of Two-Tier Coding Scheme

Input: Parent population, P_{parent}; Fitness function, f; Population size, N; Mutation probability, P_m; Generation probability, P_g; Crossover probability, P_c.

Output: the parents population.

Get the fitness of DNAs in parent population by the fitness function.
Get the median quantile f_median of fitness in the parent population.
Train IVAE using the DNA whose fitness greater than f_median.
Map the DNAs in parent population to mRNAs in latent space using IVAE encoder.
Apply the crossover operator with probability P_c.
Apply the mutation operator with probability P_m.
Apply the generation operator with probability P_g.
Map the mRNAs in latent space to DNA in initial space using IVAE decoder.
Store NDAs in the offspring population P_{gen+1}.
return the offspring population.

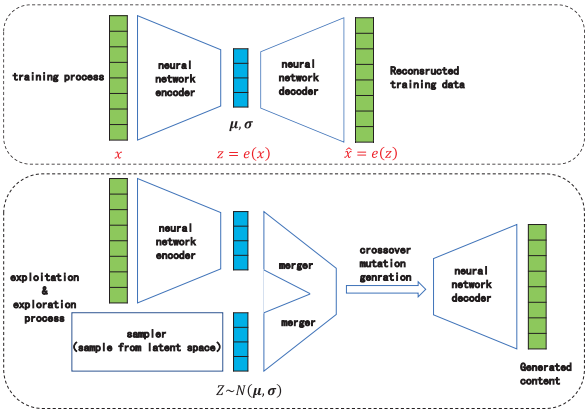


FIGURE 3. Iterative variation autoencoder.

each generation. The VAE is a generative model and is called autoencoder only because the final training objective is training a encoder and a decoder, which is resembels a traditional autoencoder. Let x denote the high-dimensional DNA code, and z denote the mRNA code in the following descriptions. The encoder neural network defines the approximate posterior distribution $q(z|x)$. This encoder network can mimic the process of stage-specific gene expression, which means that the information contained in the DNA is compressed and only the most valuable information can flow to the mRNA. The input layer size of the encoder neural network is the number of DNA coding bits. The output layer size of this neural network is the number of mRNA coding bits, which is the dimensionality of the latent space. The decoder neural network defines the conditional distribution of the observation $p(\hat{x}|z)$, which can take a latent sample z as input and output the parameters $\hat{x}$ related to z, which means reverse transcript mRNA to DNA. The input layer size of the decoder neural network is equal to the number of mRNA coding bits. The output layer size of this neural network is the number of DNA coding bits.

$$\begin{aligned} loss &= |x-\hat{x}|^2 + KL[N(\mu_x, \sigma_x), N(0,1)] \\ &= |x-d(z)|^2 + KL[N(\mu_x, \sigma_x), N(0,1)] \end{aligned} \quad (1)$$

The IVAE's encoding distribution will be regularized during each generation in the genetic algorithm. The loss function of VAE is shown in the equation 1. This loss function consists of two parts. The first part is a reconstruction term, measuring the degree of difference between the input data and the output reconstructed data. By this term, the IVAE can make the encoding network and decoding network as performant as possible. The second part is a regularization term, measuring the regularization of the latent space. By this term, the IVAE can regularize the organization of the latent space by making the distributions returned by the encoder close to a standard normal distribution [27]. The individuals with high fitness in each generation are selected as the training set for the IVAE. The individuals generated with VAE are less affected by individuals with poor fitness. This way ensure that VAE's latent space has good properties, which allow us to generate some new individuals and increase potential active transfer. It is worth noting that the IVAE in this two-tier coding scheme does not just serve as a generative model, but we can also compress the high-dimensional DNA information into low-dimensional and distributed mRNA information with the help of the IVAE encoder, and perform the traditional genetic algorithm operators in this low-dimensional latent space.

2) PROBABILISTIC FRAMEWORKS OF DNA AND mRNA

To describe our two-tier coding scheme, we define a probabilistic model. We denote by x the variable that represents DNA codes, denote by $\hat{x}$ the generated DNA codes and assume $\hat{x}$ is decoded by VAE decoder from a latent variable z which denote mRNA code. Thus, for each data point, the following two steps generative process is assumed in Fig. 4. Firstly, a latent representation z is generate by crossover operator or mutation operator using z_1 and z_2 encoded by DNA codes or sampled from the prior distribution $p(z)$ in latent space. Secondly, the generated DNA code $\hat{x}$ is decoded by the VAE decoder which can be expressed by a conditional likelihood distribution $q(x|z)$. In latent space, each variable z_i is considered to be seperable and independent on each other.

As a result, the prior distribution $p(z)$ can be computed as follows:

$$P(z) = \prod_{i=1}^{D} \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{(z_i-\mu_i)^2}{2\sigma_i}} \tag{2}$$

where μ_i and σ_i are the mean value and the variance of the ith latent variables trained by IVAE respectively.

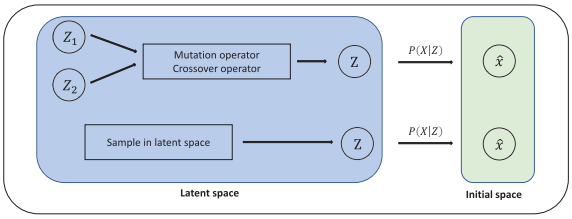


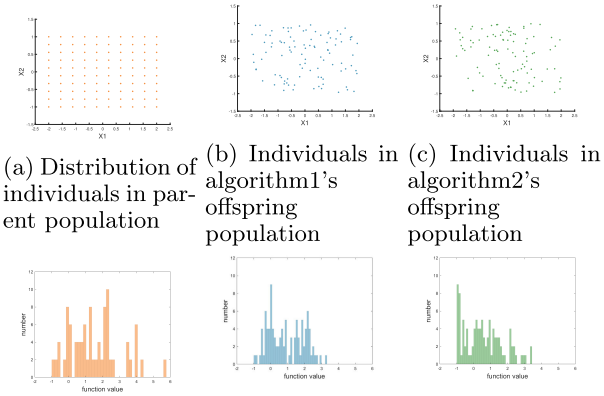
FIGURE 4. Probabilistic model of the NDA generation process.

To visualize the role of the two-tier coding scheme, we use an elite genetic algorithm coded by binary coding and an elite genetic algorithm coded by the two-tier coding scheme to find the minimum value of a six-hump camel function. The formula of this function is as in (3), shown at the bottom of the page. To be fair, the parent populations of both algorithms are composed of individuals uniformly distributed in the search space. The only difference of these two algorithms is the encoding method. We compared the mean and minimum fitness values and distribution of individuals in the offspring populations of these two algorithms. For the sake of description, we call the elite genetic algorithm coded by binary coding as algorithm 1, and call the elite genetic algorithm coded by the two-tier coding scheme as algorithm 2. The population size, crossover probability, mutation probability, and generation probability are set as 100, 0.8, 0.2, and 0.3. In order to visualize the distribution of individuals in the latent space, the dimension of the latent space is set to 2 in this section. The comparison results of Algorithm 1 and Algorithm 2 are shown in Tbale 1 and Fig 5a, 5b, 5c, 5d, 5e, 5f.

TABLE 1. Statistic of individuals' function value.

	mean value	minimum value
parent population	1.5071	-0.9363
offspring population of algorithm 1	0.9410	-0.9716
offspring population of algorithm 2	0.3660	-1.0188

From Table 1 we can see that the mean and minimum values of the objective function of individuals in the subpopulation of Algorithm 2 are lower than those of Algorithm 1, which indicates that the algorithm with the two-tier coding



(a) Distribution of individuals in parent population
(b) Individuals in algorithm1's offspring population
(c) Individuals in algorithm2's offspring population
(d) Histogram of individuals in parent population
(e) Histogram of algorithm1's offspring population
(f) Histogram of algorithm2's offspring population

FIGURE 5. Distribution of individuals.

scheme has a stronger optimization ability. The Fig 5a shows the initial distribution of individuals in algorithm1's parent population, which is a uniform distribution. The Fig 5b and Fig 5c show the distribution of individuals in offspring populations. These offspring populations are after one iteration of algorithm1 and algorithm2 respectively. The individuals in these offspring populations exhibit aggregation toward points with low function values. The Fig 5d, 5e, 5f show the histograms of individuals's function values in parent population and offspring populations respectively. The shape of Fig 5d has no clear pattern, so we can describe this distribution as "random". The shape of Fig 5e and Fig 5f have a "tail" on the right side of the distribution which is called right skewed, and the offspring population of the algorithm with the two-tier coding scheme has a heavier right skewed, which indicates that the algorithm with the two-tier coding scheme has a stronger optimization ability. With the two-tier coding scheme, the genetic algorithm can find the optimal solution faster.

B. FILTER OPERATOR

1) MECHANISM OF A FILTER OPERATOR

The select operator is an important operator in genetic algorithms. The convergence rate of GA depends upon the selecting pressure. Some well-known select operators like tournament, roulette wheel, Boltzmann, use the objective function as the fitness function, to measure of individuals' quality, then determine whether an individual will participate the process generating offspring population or not. These select operators use the principle of natural selection from biological evolution. Through this mechanism, individuals

$$f(x_1, x_2) = \left(4 - 2.1\, x_1^2 + \frac{x_1^4}{3}\right) x_1^2 + x_1 x_2 + \left(-4 + 4x_2^2\right) x_2^2 \quad x_1 \in [-2, 2], x_2 \in [-1, 1] \tag{3}$$

with poor fitness have little chance to participate in the generation of the child population. But in the real world, poorly behaved individuals like Down's and autistic, can show talent in some areas [28]. This phenomenon indicates that in the population of GA, although some individuals have low fitness, they may have great potential to generate individuals with high fitness in the next generation.

In this subsection, we introduce an improved select operator named filter operator. This operator measures the quality of an individual in two dimensions. The first is traditional fitness function, and the second is the degree of outliers of the individual in the latent space.

Individual fitness can be obtained by calculating the objective function. Below, we will give the measure of individual outlier degree. In the previous subsection, we use population data to train IVAE in each generation. The encoder neural network can return a multivariate gaussian distribution over the latent space by a regularization term in loss function. The covariance matrix of this distibution is a diagonal matrix. After training IVAE, we put the DNA information into the encoder to get the corresponding mRNA in latent space. Every dimension i in the latent space following a Gaussian distribution with mean $\boldsymbol{\mu_i}$ and standard deviation $\boldsymbol{\sigma_i}$. These parameters can get by training IVAE. A simple statistical tool for outlier detection is Z-score, which indicates how far the sample is from the global mean. measure the outlier of mRNA in latent space. We extend Z-score to higher dimensions, compute the Z-score of each dimension separately, and sum the squares of Z-score of each dimension. Outlier score is expressed as (4):

$$d(z) = \sum_{i=1}^{n} \left(\frac{z_i - \mu_i}{\sigma_i}\right)^2 \tag{4}$$

where $\boldsymbol{\mu}$ is the mean vector of the multivariate Gaussian distributions and $\boldsymbol{\sigma}$ is the diagonal of the covariance matrix, respectively, z is a mRNA, and z_i is the value of z in i_{th} dimension.

As the output of our encoder is regularity, the outlier of mRNA is an extreme value that does not follow the norm. The outlier with high fitness can help us learn some unexpected knowledge, which means exploration. These good outliers can point out the new search direction for our genetic algorithm. But the outlier with low fitness is often seen as a problem rather than a help. These bad outliers can affect external validity and cause problems along with the process of population evolution. So, our filter operator needs to select two types of individuals from the parent population. The first type is individuals with high fitness and low outliers in the latent space, which is an exploitation process. The second type is outlier individuals whose fitness are not too bad, which is an exploration process. To maintain the size of the population, the insufficient fraction is generated from the latent space using IVAE. These generated offspring inherit the characteristics of the parent population. Algorithm 2 summarized the general procedure of a filter operator.

2) EXAMPLE OF A FILTER OPERATOR

In this section, we continue to use the optimization problem 3 and algorithm 2 from Section III-A2 as an example. The algorithm 2 is an elite genetic algorithm coded by the two-tier coding scheme. We embed the filter operator into this algorithm. The advantage of this operator is illustrated by comparing the distribution changes of individuals in the original search space and the latent space shown in Fig 6. In contrast to the case of the latent space without the filtering operator, the distribution of population filtered by filter operator is tighter in the latent space, this population consists of exploration sub-populations, exploitation sub-populations, and generation sub-populations respectively. The mean and minimum objective functions values of the individuals in the offspring populations of the algorithm with the filtering operator is lower as recorded in Table 2.

TABLE 2. Statistic of individuals' object function value.

	mean value	minimum value
parent population	1.5071	-0.9363
offspring population of algorithm 2 without filter operator	0.5249	-1.0160
offspring population of algorithm 2 with filter operator	0.1100	-1.0234

C. THE STAGE-SPECIFIC GENE EXPRESSION FRAMEWORK

The stage-specific gene expression framework (SGEF) is proposed for optimizing the speed, efficiency, and quality of genetic algorithms. SGEF introduces a two-tier coding scheme and a population filter operator. To clarify the running of the SGEF, its complete flowchart is provided in Fig 7. The pink frame stands for a general genetic algorithm. The violet frame stands for the two-tier coding scheme. The blue frame stands for the population filter operator. In each iteration of a general genetic algorithm, the parent population will train an IVAE, and then the trained IVAE will map the DNA input to mRNA output. A filter operator will tradeoff the fitness and potential of these mRNA. The filtered mRNAs will mate and produce better offspring by the select operator, crossover operator, and mutation operator. These offspring which inherit the characteristics of the parent individuals will be added to the child population. The child population is the parent population in the next iteration of this genetic algorithm. The red line indicates the interface of SGEF. Our stage-specific gene expression framework can embed into any genetic algorithm easily.

IV. EXPERIMENTAL RESULTS

In our experimental studys, we verify the performance of SGEF using 17 well-known benchmark and two practical applications from three perspectives. The first is getting parameter settings of SGEF using two single-objective optimization problems; The second is numericla experiment. 15 optimization problems including single-objective optimization problems and multi-objective optimization problems are employed to investigate the effectiveness and generality of SGEF; The third is to verify the feasibility of

Algorithm 2 Pseudo-Code of Filter Operator

Input: Parent population, P_{parent}; Objective function, f; Population size, N; Proportion of the exploitation individuals, $p_{exploitation}$; Proportion of the exploration individuals, $p_{exploration}$; Proportion of the generated individuals, $p_{generation}$.
Output: the filtered parents population.

```
Evaluate the solutions in parent population by calculate the objective function.
Train IVAE using parent population.
Map the solutions in parent population to latent space using IVAE encoder.
Evaluate the outlier of solutions in latent space.
counter = 0.
Exploitation population= {}.
while (counter ≤ integer(N × p_exploitation)) do
    Select a high fitness and low outlier solution without replacement.
    Put selected solution in exploitation population.
    counter = counter + 1.
end while
counter = 0.
Exploration population= {}.
while (counter ≤ integer(N × p_exploration)) do
    Select a high fitness and high outlier solution without replacement.
    Put selected solution in exploration population.
    counter = counter + 1.
end while
if p_exploitation + p_exploration ≤ 1 then
    counter = 0.
    generative population = {}.
    while (counter ≤ integer(N × p_generation)) do
        generate a solution from the latent space using IVAE.
        put generated solution in generation population.
        counter = counter + 1.
    end while
end if
Merge exploration population, exploitation population and generative population.
return the merged population.
```

SGEF in practical applications. The algorithms were programed in Python 3.7 version and executed on computation environment of Intel(R) Core(TM) i5-9400F CPU @ 2.90GHz 2.90 GHz, NVIDIA GeForce RTX 2060, 16GB RAM and 64-bit operating system.

A. INFLUENCE OF THE PARAMETER SETTINGS

Parameter settings can significantly influence the performance of a genetic algorithm. In SGEF, the main parameters include exploitation proportion ($p_{exploitation}$), exploration proportion ($p_{exploration}$), and generation proportion ($p_{generation}$). The range of values for these parameters are as follwings: $p_{exploitation} \in \{0, 0.25, 0.5, 0.75, 1\}$, $p_{exploration} \in \{0, 0.25, 0.5, 0.75, 1\}$, and $p_{generation} \in \{0, 0.25, 0.5, 0.75, 1\}$. We use limited full factorial design approach to analysis parameters sensitivity. Besides the parameters of the selected algorithms, different scenarios of parameter setting are presented in Table 4. The test functions are 2 dimensional rosenbrock function f_1 [29] and michalewicz function f_2 [30] selected in appendix Table 16.

In this subsection, we embed SGEF into three single-objective optimization genetic algorithms, namely, SGA, EGA, and StudGA. These algorithms form three new algorithms, namely, SGEF-SGA, SGEF-EGA, SGEF-StudGA. The performance of the original algorithms are used as the control group, and the reconstructed algorithms are used as the experimental group. Thus, three comparisons of SGEF-SGA versus SGA, SGEF-EGA versus EGA, and SGEF-StudGA versus StudGA were conducted. To help these algorithms achieve the best results, some generic parameter settings of selected algorithms are mentioned in Table 3, which is the same as suggested in their original references [31], [32], [33], [34], [35], [36]. When the population size is set too large, all six algorithms converge quickly with insignificant differences. To verify the acceleration effect of SGEF on the algorithms, the population size is set as 100 in this section.

Since a genetic algorithm is stochastic in nature, the running result of this algorithm is also not fixed. Therefore, we executed it 10 times for each parameter setting scenario.

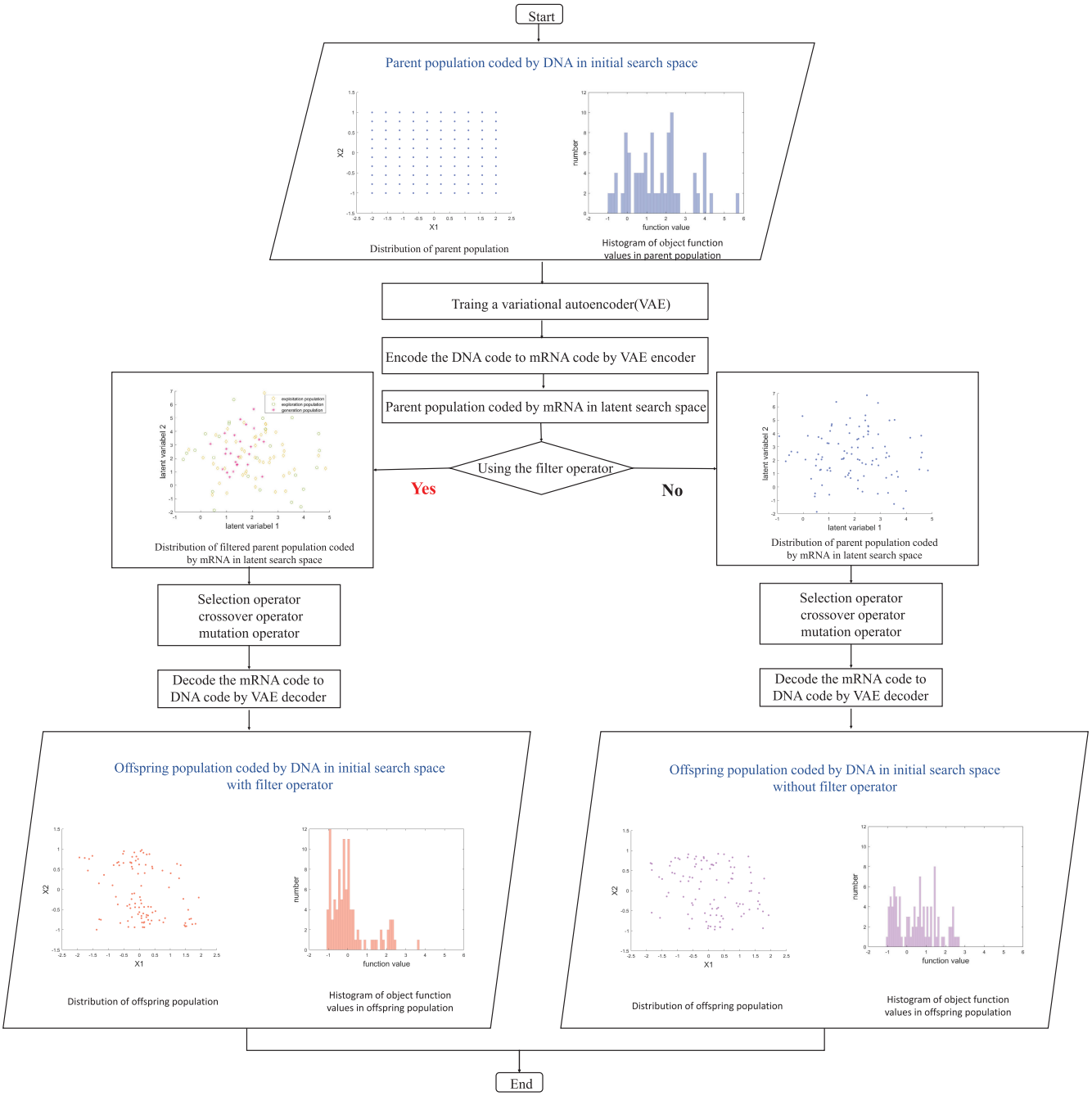


FIGURE 6. Flowchart of the filter operator.

In each calculation, when the change of the fitness function value corresponding to the optimal solution does not exceed 0.001 during three consecutive iterations, the iteration will end. Their mean results are recorded in Table 5 and Table 6. The data in Table 5 and Table 6 have two dependent variables, which are parameter setting scenarios and the type of algorithm. We use multivariate analysis of variance (MANOVA) to compare multivariate sample means. The MANOVA results are recorded in Table 7.

In our preliminary testing, based on the p-values from Tables 7, it is clear that there is an association between parameters' scenarios and type of algorithm. By simply calculating the average value of each dimension, we can also conclude that the proposed framework in this paper can improve the efficiency of the existing algorithms. It is clear that scenario number 11 is the best value for these parameters. With parameter setting as $p_{exploitation} = 0.5$, $p_{exploration} = 0.25$, and $p_{generated} = 0.25$. The performance of algorithms based on SGEF can achieve the best results.

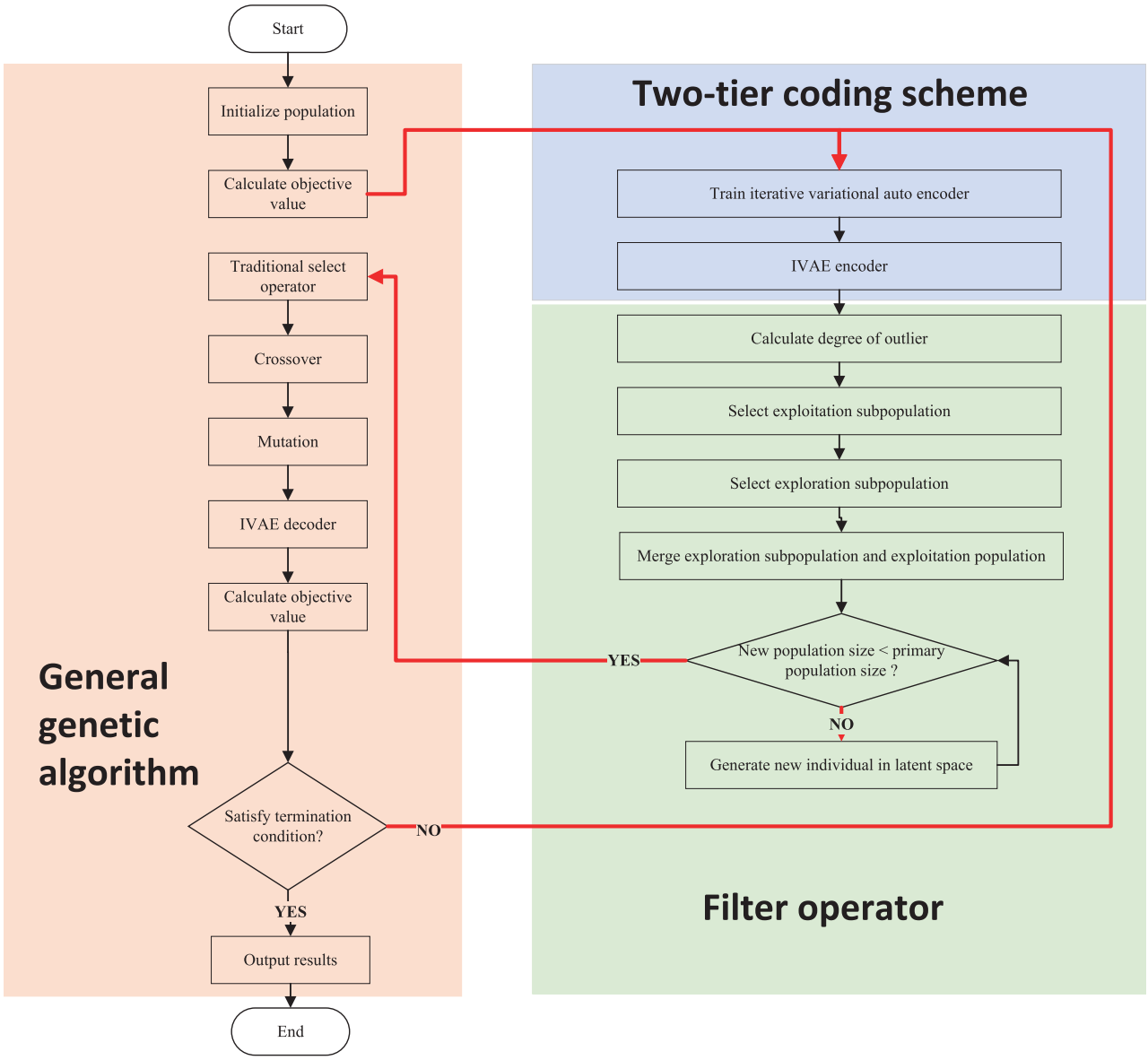


FIGURE 7. SGEF flowchart.

To examine SGEF's ability to accelerate convergence speed, we executed 10 times with each algorithm and each test function using parameter settings based on scenario 11 in table 4. Table 9 records the iteration numbers of each algorithm. Table 8 records the mean time cost per iteration for each algorithm. As can be seen from the table 8, the time required for each iteration is reduced because the SGEF embedded in these algorithms can accelerate the computation with the help of GPU.

For further proof the accelerating ability of the SGEF, we first create boxplots to visualize the distribution of iteration numbers for each group. In Fig 8a and Fig 8b we can intuitively see the distribution of iteration numbers of each algorithm on f_1 and f_2 based on five summary number (minimum, first quartile, median, third quartile and maximum). By comparing the vertical lines in each box, we can determine algorithms using SGEF have a lower median value. By comparing the length of each box and the interquartile range, we can also determine that the algorithms using SGEF are less spread out and negatively skewed compared to the initial algorithms.

To be more convincing, we analze the data recorded in Table 9 using the Welch-Satterthwaite's T-test. Welch-Satterthwaite's T-test is used to compare the means between two independent groups when it is not assumed that the two groups have equal variances. The t test statistics and probability values (p-value) are recorded in Table 10.

As we can see from Table 10, five of six probability values of the Welch-Satterthwaite's T-tests are less than 0.05. We can reject the null hypothesis and conclude that there is a

TABLE 3. Parameter setting of selected algorithms.

Algorithms	Paremeters	
SGA	Crossover probability	0.8
	Mutation probability	0.6
EGA	Crossover probability	0.8
	Mutation probability	0.2
	Elite individuals	5
StudGA	Crossover probability	1
	Mutation probability	0.003
MOEA/D	Crossover probability	1
	Mutation probability	$\frac{1}{n}$
	Distribution index (n_c)	30
	Distribution index (n_m)	30
NSGA-III	Crossover probability	1
	Mutation probability	$\frac{1}{n}$
	Distribution index (n_c)	40
	Distribution index (n_m)	30
RVEA	Crossover probability	1
	Penalty function controller(α)	2
	Frequency F_c	0.1
	Mutation probability	$\frac{1}{n}$
	Distribution index (n_c)	30
	Distribution index (n_m)	30

TABLE 4. Different parameters setting scenarios of SGEF.

Scenarios		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Parameter values	Pexploitation	0	0	0	0	0	0.25	0.25	0.25	0.25	0.5	0.5	0.5	0.75	0.75	1
	Pexploration	0	0.25	0.5	0.75	1	0	0.25	0.5	0.75	0	0.25	0.5	0	0.25	0
	Pgeneration	1	0.75	0.5	0.25	0	0.75	0.5	0.25	0	0.5	0.25	0	0.25	0	0

TABLE 5. Experiment results of f_1 for SGEF under different scenarios.

Scenarios	f_1					
	SGA	SGEF-SGA	EGA	SGEF-SGA	StudGA	SGEF-StudGA
1	2.234E-01	7.783E-01	5.243E-02	2.384E-01	1.922E-01	4.153E-01
2	2.234E-01	5.483E-01	5.243E-02	5.257E-01	1.223E-02	2.116E-01
3	2.234E-01	1.682E-02	5.243E-02	2.620E-02	1.223E-02	1.110E-02
4	2.234E-01	4.790E-02	5.243E-02	1.310E-02	1.223E-02	9.230E-03
5	2.234E-01	1.746E-01	5.243E-02	1.682E-01	1.223E-02	2.734E-01
6	2.234E-01	5.560E-02	5.243E-02	1.420E-02	1.223E-02	7.190E-03
7	2.234E-01	6.990E-02	5.243E-02	2.270E-02	1.223E-02	7.620E-03
8	2.234E-01	6.780E-02	5.243E-02	2.320E-02	1.223E-02	7.810E-03
9	2.234E-01	7.210E-02	5.243E-02	2.145E-02	1.223E-02	7.150E-03
10	2.234E-01	5.670E-02	5.243E-02	1.843E-02	1.223E-02	6.940E-03
11	2.234E-01	3.120E-02	5.243E-02	8.380E-03	1.223E-02	5.290E-03
12	2.234E-01	3.660E-02	5.243E-02	9.260E-03	1.223E-02	5.220E-03
13	2.234E-01	8.660E-02	5.243E-02	1.250E-02	1.223E-02	6.280E-03
14	2.234E-01	8.360E-02	5.243E-02	1.620E-02	1.223E-02	6.930E-03
15	2.234E-01	1.573E-01	5.243E-02	1.590E-02	1.223E-02	8.230E-03

TABLE 6. Experiment results of f_2 for SGEF under different scenarios.

Scenarios	f_2					
	SGA	SGEF-SGA	EGA	SGEF-SGA	StudGA	SGEF-StudGA
1	-1.635E+00	-7.892E-01	-1.793E+00	1.092E+00	-1.801E+00	1.013E+00
2	-1.635E+00	-1.935E-01	-1.793E+00	-2.194E-01	-1.801E+00	-3.013E-01
3	-1.635E+00	-1.784E+00	-1.793E+00	-1.798E+00	-1.801E+00	-1.801E+00
4	-1.635E+00	-1.801E+00	-1.793E+00	-1.800E+00	-1.801E+00	-1.801E+00
5	-1.635E+00	-1.101E+00	-1.793E+00	-1.300E+00	-1.801E+00	-9.301E-01
6	-1.635E+00	-1.801E+00	-1.793E+00	-1.800E+00	-1.801E+00	-1.801E+00
7	-1.635E+00	-1.792E+00	-1.793E+00	-1.801E+00	-1.801E+00	-1.801E+00
8	-1.635E+00	-1.796E+00	-1.793E+00	-1.799E+00	-1.801E+00	-1.801E+00
9	-1.635E+00	-1.733E+00	-1.793E+00	-1.801E+00	-1.801E+00	-1.801E+00
10	-1.635E+00	-1.799E+00	-1.793E+00	-1.801E+00	-1.801E+00	-1.801E+00
11	-1.635E+00	-1.801E+00	-1.793E+00	-1.801E+00	-1.801E+00	-1.801E+00
12	-1.635E+00	-1.801E+00	-1.793E+00	-1.801E+00	-1.801E+00	-1.801E+00
13	-1.635E+00	-1.769E+00	-1.793E+00	-1.801E+00	-1.801E+00	-1.801E+00
14	-1.635E+00	-1.801E+00	-1.793E+00	-1.800E+00	-1.801E+00	-1.801E+00
15	-1.635E+00	-1.780E+00	-1.793E+00	-1.731E+00	-1.801E+00	-1.801E+00

statistically significant difference in mean iteration number when SGEF-SGA versus SGA, SGEF- EGA versus EGA, and SGEF-StudGA versus StudGA. The convergence speed of the algorithms using SGEF is generally faster than original algorithms. Therefore, it is reasonable to conclude that our SGEF has good search performance and convergence.

TABLE 7. Multivariate analysis of variance results.

		DF	Sum Sq	Mean Sq	F value	P value
f_1	Scenarios	14	0.5565	0.03975	4.382	1.55E-05
	Algorithms	5	0.4156	0.08312	9.162	9.68E-07
	Residuals	70	0.6351	0.00907		
f_2	Scenarios	14	0.5565	0.03975	4.382	1.55E-05
	Algorithms	5	0.4156	0.08312	9.162	9.68E-07
	Residuals	70	0.6351	0.00907		

TABLE 8. Mean time required per iteration for each algorithm.

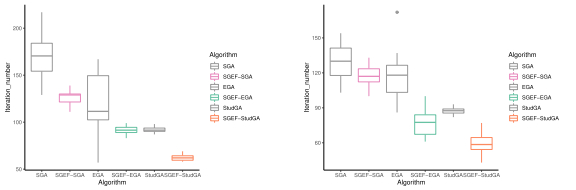
Optimization function	SGA	SGEF-SGA	EGA	SGEF-EGA	StudGA	SGEF-StudGA
f_1	3.238E-03s	3.122E-03s	3.292E-03s	3.249E-03s	3.836E-03s	3.501E-03s
f_2	5.387E-03s	4.432E-03s	5.820E-03s	5.729E-03s	6.332E-03s	6.163E-03s

TABLE 9. Iteration numbers to optimize f_1 and f_2.

Test	f_1						f_2					
	SGA	SGEF-SGA	EGA	SGEF-EGA	StudGA	SGEF-StudGA	SGA	SGEF-SGA	EGA	SGEF-EGA	StudGA	SGEF-StudGA
1	129	130	153	89	95	65	120	131	101	87	87	77
2	217	111	102	88	94	62	150	117	125	61	85	63
3	171	130	104	93	88	62	130	116	137	76	82	67
4	155	139	111	93	93	58	154	133	172	66	90	50
5	170	128	57	83	91	67	117	104	110	79	89	54
6	152	133	167	90	91	62	103	122	127	65	88	55
7	154	130	139	89	90	59	103	124	121	81	83	65
8	186	120	112	95	92	69	145	111	86	85	93	55
9	184	126	165	99	87	61	130	117	115	100	89	62
10	184	115	88	99	98	59	130	100	89	71	87	43

TABLE 10. Results of the Welch-Satterthwaite's T-test.

Result	f_1			f_2		
	SGA versus SGEF-SGA	EGA versus SGEF-EGA	StudGA versus SGEF-StudGA	SGA versus SGEF-SGA	EGA versus SGEF-EGA	StudGA versus SGEF-StudGA
t test statistic	8.2007	50.095	19.162	2.843	4.4395	8.7212
probability value	4.405E-03	2.569E-06	2.366E-13	1.355E-01	3.684E-02	2.731E-06



(a) Box plot of iteration number on f_1 (b) Box plot of iteration number on f_2

FIGURE 8. Box plot of iteration number on f_1 and f_2.

B. NUMERICAL EXPERIMENT

1) TEST PROBLEMS AND TEST ALGORITHMS

To verify the effectiveness and generality of SGEF, fiften well-known benchmark constrained functions are selected, including eight single-objective optimization problems and seven multi-objective optimization problems. The details of eight single-objective optimization problems are f_3, f_4, f_5, f_6, f_7, f_8, f_9, and f_{10} recorded in appendix Table 16. The seven multi-objective optimization problems are selected from DTLZ test suite [37], which have 20 decision variables and 3 objective functions.

We embed SGEF into six genetic algorithms, including three traditional single-objective optimization genetic algorithms named simple genetic algorithm(SGA) [32], elite genetic algorithm(EGA) [36], stud genetic algorithm (StudGA) [31] and three state-of-the-art multiple-objective optimization genetic algorithms named reference

vector-guided evolutionary algorithm (RVEA) [33], non-dominated sorting genetic algorithm III (NSGA-III) [34], multi-objective evolutionary algorithm based on decomposition(MOEA/D) [35]. The reconstructed algorithms are SGEF-SGA, SGEF-EGA, SGEF-StudGA, SGEF-RVEA, SGEF-NSGA-III, SGEF- MOEA/D.

2) EFFECTIVENESS OF SGEF ON SINGLE-OBJECTIVE OPTIMIZATION PROBLEMS

In this subsection, we verify the robustness and portability of SGEF on ten single-objective optimization problems. These test problems have different test function (e.g., linear, cubic, quadradic, and nonlinear), which are discribed in appendix Table 16. we embed SGEF into three single-objective optimization genetic algorithms, namely, SGA, EGA, and StudGA. These algorithms form three new algorithms, namely, SGEF-SGA, SGEF-EGA, SGEF-StudGA. The performance of the original algorithms are used as the control group, and the reconstructed algorithms are used as the experimental group. Thus, three comparisons of SGEF-SGA versus SGA, SGEF- EGA versus EGA, and SGEF-StudGA versus StudGA were conducted. Since a genetic algorithm is stochastic in nature, the running result of this algorithm is also not fixed. Therefore, we executed it 10 times for each algorithm and optimization problem. The population size is set at 200, and the maximum iterations is set at 300. Other parameters are set as Table 3. In each optimization, when the change of the fitness function value corresponding to the optimal solution does not exceed 0.0001 during three consecutive iterations, the iteration will end. Their mean optimal values are recorded in Table 11 and their mean convergence time are recorded in Table 12.

TABLE 11. Mean optimal value of 8 test functions for different algorithms.

Problem	SGA	SGEF-SGA	EGA	SGEF-EGA	StudGA	SGEF-StudGA
f_3	5.0310E+00	5.0178E+00	5.0183E+00	5.0104E+00	5.0035E+00	5.0011E+00
f_4	3.3800E-02	1.5600E-02	9.0000E-03	7.3000E-03	8.4000E-03	3.6000E-03
f_5	-1.8410E+02	-1.8566E+02	-1.8601E+02	-1.8613E+02	-1.8618E+02	-1.8649E+02
f_6	3.7830E-01	2.9570E-01	2.3290E-01	2.2700E-01	1.5450E-01	1.0510E-01
f_7	-6.8120E+03	-6.9210E+03	-6.9381E+03	-6.9428E+03	-6.9387E+03	-6.9522E+03
f_8	6.2000E-03	3.2000E-03	3.0290E-03	3.0120E-03	2.9140E-03	2.6789E-03
f_9	4.0085E+00	4.0076E+00	4.0057E+00	4.0034E+00	4.0019E+00	4.0013E+00
f_{10}	1.5328E+00	1.4994E+00	1.4915E+00	1.4938E+00	1.4914E+00	1.4887E+00

The results show that SGEF can accelerate the calculation speed and enhance the search ability of single-objective optimization genetic algorithms significantly. In addition, we calculate the absolute value of the difference between the optimal values in the table 11 and the ture optimal values recorded in appendix Table 16. The precision of algorithms using SGEF is 32.60%, 17.61%, and 37.76% higher than that of original algorithms, respectively, and the average running time of algorithms using SGEF is 14.20%, 24.96% and 15.03% faster than that of original algorithms, respectively. Therefore, the SGEF can improve the convergence speed and search ability of single-objective optimization genetic algorithms, which verifies the effectiveness of the SGEF.

TABLE 12. Mean time costing of 8 test functions for different algorithms.

Problem	Mean time costing /s					
	SGA	SGEF-SGA	EGA	SGEF-EGA	StudGA	SGEF-StudGA
f_3	1.1252	0.9756	0.8401	0.8009	0.5732	0.5134
f_4	0.9823	0.7851	0.6756	0.5871	0.4273	0.4192
f_5	20.3415	18.4580	16.0995	14.4681	13.8261	11.2350
f_6	3.2532	2.9789	1.9560	1.7216	0.8363	0.7624
f_7	25.3467	21.2348	20.5290	19.2794	19.2698	16.2395
f_8	2.1435	1.9920	1.2906	1.3137	0.7143	0.8234
f_9	1.4234	1.3653	0.8755	0.9114	0.3812	0.3173
f_{10}	3.5624	2.1246	2.8458	2.2876	2.6936	2.5915
total	58.1782	49.9143	47.3312	35.5164	38.7218	32.9017

3) EFFECTIVENESS OF SGEF ON MULTIPLE-OBJECTIVE OPTIMIZATION PROBLEMS

We further verify the robustness and portability of SGEF on seven well-studied test problems, namely DTLZ1 to DTLZ7 [37], which are three-objective problems. Three multiple-objective optimization genetic algorithms, namely, RVEA, NSGA-III, and MOEA/D, are embeded in SGEF and form three new algorithms, namely, SGEF-RVEA, SGEF-NSGA-III, and SGEF-MOEA/D. In this subsection, the original algorithms are used as the control group, and the algorithms using SGEF are used as the experimental group. Thus, three comparisons of SGEF-RVEA versus RVEA, SGEF-NSGA-III versus NSGA-III, and SGEF-MOEA/D versus MOEA/D were conducted. The population size is set at 400, and the maximum iterations is set at 300. Other parameters are set as Table 3.

We use DTLZ problems which have 20 decision variables and 3 objective functions as representative cases, and use inverted generational distance (IGD) [38] to measure the performance of these algorithms. IGD is a metric which can reflect the convergence and diversity of algorithms' results. A smaller IGD represents the obtained solution set has better performance on both diversity and convergence. Let PF_{ture} be a set of evenly distributed point sampled from pareto frontier. Let PF_{known} be an approximate set obtained by algorithms. Then, the $IDG(PF_{ture}, PF_{known})$ are defined as (5).

$$IDG(PF_{ture}, PF_{known}) = \frac{\sum_{i=1}^{n} |d_i|}{n} \tag{5}$$

Here d_i represents Euclidean distance from on point of PF_{ture} to its nearest points of PF_{known}. The performances of all algorithms are compared on all the DTLZ test problems. To obtain relatively stable results, we executed it 10 times for each algorithm and optimization problem.

The performance of all the multi-objective genetic algorithms are compared on the DTLZ test problems which have 20 dimensions. Their IGD values under 10 independent runs on the DTLZ test problems are given in Table 13. Algorithms' mean IGD values versus the iteration number are plotted in Fig 9. Algorithms' mean convergence time are recorded in Table 14.

As observed from Fig 9, the vertical coordinates represent the average distance from the solution set to the pareto frontier, while the red lines with circle markers representing Algorithms using SGEF, the blue lines with star markers representing original algorithms. In terms of the number of

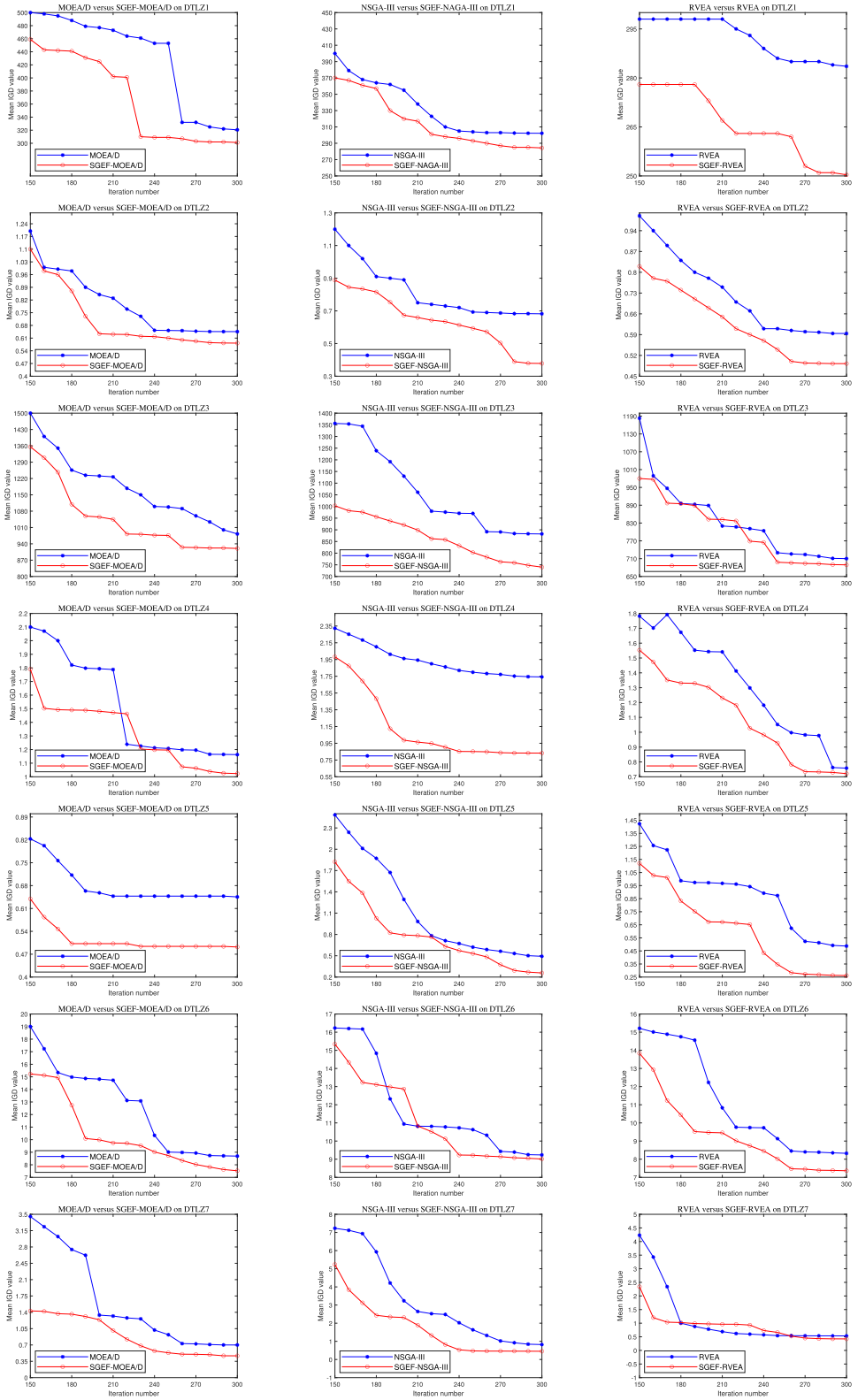


FIGURE 9. Convergence curves of competitor algorithms on DTLZ with 20 dimensions.

iterations required to reach convergence, algorithms usign SGEF can converge faster than original algorithms. The data recorded in the table 13 also shows that the IGD values of Algorithms using SGEF are also generally lower than those of original algorithms. This implies that algorithms using SGEF have a better search capability.

TABLE 13. Mean IGD of DTLZ test suite for different algorithms.

Problem	MOEA/D	SGEF- MOEA/D	NSGA-III	SGEF-NSGA-III	RVEA	SGEF-RVEA
DTLZ1	3.206E+02	3.122E+02	3.022E+02	2.843E+02	2.836E+02	2.504E+02
DTLZ2	6.457E-01	5.823E-01	6.820E-01	3.782E-01	5.939E-01	4.924E-01
DTLZ3	9.824E+02	9.203E+02	8.823E+02	7.402E+02	7.101E+02	6.892E+02
DTLZ4	1.163E+00	1.023E+00	1.742E+00	8.325E-01	9.583E-01	7.213E-01
DTLZ5	6.454E-01	4.920E-01	4.923E-01	2.581E-01	4.872E-01	2.617E-01
DTLZ6	8.692E+00	7.524E+00	9.235E+00	9.012E+00	8.324E+00	7.364E+00
DTLZ7	6.999E-01	4.674E-01	8.230E-01	4.523E-01	5.334E-01	4.230E-01

TABLE 14. Mean time costing of DTLZ test suite for different algorithms.

Problem	Mean time costing /s					
	MOEA/D	SGEF- MOEA/D	NSGA-III	SGEF-NSGA-III	RVEA	SGEF-RVEA
DTLZ1	34.095	24.21746667	4.838166667	4.523033333	5.1698	5.0058
DTLZ2	27.3176	22.11885	5.74848	5.46157	4.13352	4.3095333
DTLZ3	34.337	29.85301333	5.56752	5.5189	4.85754	4.52988
DTLZ4	32.2812	34.19003333	6.5589	4.771853333	5.17766	4.64607
DTLZ5	33.26203333	28.1096	9.5079	5.13764	4.1873	3.41262
DTLZ6	32.19906667	35.438	7.627293333	5.14989	4.7973	3.630466667
DTLZ7	36.293	32.66573333	6.505053333	4.56141	3.77469	3.78702

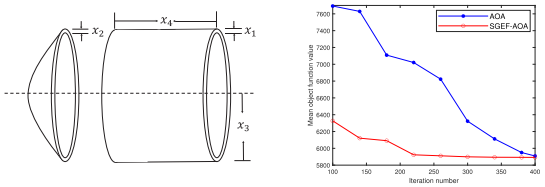
TABLE 15. Wilcoxon test results for time costing.

Problem	MOEA/D versus SGEF-MOEA/D	NSGA versus SGEF-NSGA	RVEA versus SGEF-RVEA
Wilcoxon statistic	25	28	24
probability value	0.03906	0.007813	0.05469

To statistically validate the time costing results achieved by SGEF, we performe nonparametric Wilcoxon Ranksum test, as it serves to produce meaningful comparison between SGEF-RVEA and RVEA, SGEF-NSGA-III and NSGA-III, SGEF-MOEA/D and MOEA/D. The alternative hypothesis of this test is the time costing of algorithms using SGEF is shorter than that of original algorithms. The p-values for the nonparametric Wilcoxon Ranksum test are given in Table 15. According to Table 15, the time costings of algorithms using SGEF remained significantly shorter than the original algorithms for DTLZ test problems. The average running times of algorithms using SGEF are 10.09%, 24.22% and 8.64% faster than that of original algorithms, respectively. It is reasonable to conclude that our SGEF can accelerate the calculation speed and enhance the search ability of multiple-objective optimization genetic algorithms significantly.

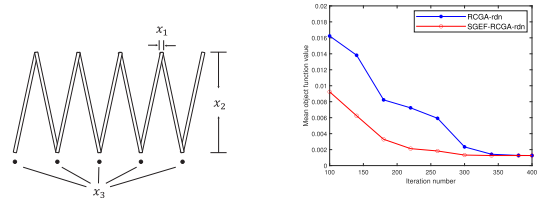
C. FEASIBILITY OF SGEF IN PRACTICAL APPLICATIONS

Genetic algorithms have been used to solve optimization problems in an effective manner. To name a few domains, Polymer design in materials [1], detection of Parkinson disease in Clinical Medicine [2], mechanical design problems in engineering [3] and Transportation energy demand forecasting in Operations Research [4]. Since we do not know the optimal Pareto front of the multi-objective optimization problems in real life, it is not possible to visually compare SGEF's promotion on practical multi-objective genetic algorithms. To verify the overall feasibility of SGEF, we test SGEF for solving two constrained single-objective engineering design problems found in appendix B and appendix C, and the descriptions of the problems are as follows.



(a) Schematic design (b) Convergence curve

FIGURE 10. Pressure vessel design problem.



(a) Schematic of tension-compression spring (b) Tension-compression spring design problem's Convergence curve

FIGURE 11. Tension-compression spring design problem.

1) PRESSURE VESSEL DESIGN PROBLEM

Hashim et al. proposed a new metaheuristic algorithm called Archimedes optimization algorithm (AOA) inspired by the physics Archimedes Principle, and use AOA to solve a hybrid constrained optimization problem [39]. The purpose of the optimization is to mimimize the cost of pressure vessel design (Figure 10a). The mathematical model is described in appendix B. The cost is optimized with the help of four design variables: the shell thickness(x_1), the head thickness(x_2), the inner radius(x_3), and the cylinder length(x_4). In this paper, we embed SGEF into AOA, and the objective function value achieved by SGEF-AOA is 5.892E+03 which is better than 5.9092E+03 acheved by original AOA. Figure 10b also suggests that SGEF-AOA have a faster convergence ability than original AOA.

2) TENSION-COMPRESSION SPRING DESIGN PROBLEM

Song et al. proposed an improved real-coded genetic algorithm (RCGA-rdn) integrating three specially designed operators, and use RCGA-rdn to solve a inequality constrained optimization problem (Figure 11a) [40]. The purpose of the optimization is to mimimize mass of a spring under the conditions of mimimum deflection, shear stress, and vibration frequency in tension-compression spring design. The desing variables are wire diameter(x_1), coil diameter(x_2), and number of active coils(x_3). The mathematical model is described in appendix C. In this paper, we embed SGEF into RCGA-rdn, although the objective function value achieved by SGEF-RCGA-rdn is 1.2612E-02 which is not significantly better than 1.2618E-02 acheved by original AOA. Figure 11b suggests that SGEF-RCGA-rdn have a faster convergence ability than original RCGA-rdn.

TABLE 16. Basic information about single-object test function.

Test function	Input domain	Dimension	Optimum
$\min f_1(\mathbf{x}) = \sum_{i=1}^{d-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	$x_1 \in [-2.048, 20.48]$ $x_2 \in [-2.048, 20.48]$	2	$f(x^*)=0$
$\min f_2(\mathbf{x}) = -\sum_{i=1}^{d}\sin\left(x_i\right)\sin^{20}\left(\frac{ix_i^2}{\pi}\right)$	$x_1 \in [o, \pi]$ $x_2 \in [o, \pi]$	2	$f(x^*)=-1.8012$
$\min f_3(x) = \left(x_1-2\right)^2+\left(x_2-1\right)^2$ s.t. $\begin{cases} x_1+x_2-2\le 0 \\ x_1^2-x_2+2\le 0 \end{cases}$	$x_1 \in [-5, 5]$ $x_2 \in [-5, 5]$	2	$f(x^*)=5$
$\min f_4(x) = \left(x_1^2+x_2-11\right)^2+\left(x_1+x_2^2-7\right)^2$ s.t $\begin{cases} \left(x_1-0.05\right)^2+\left(x_2-2.5\right)^2-484\le 0 \\ 4.84-x_1^2-\left(x_2-2.5\right)^2\le 0 \end{cases}$	$x_1 \in [0, 6]$ $x_2 \in [0, 6]$	2	$f(x^*)=0$
$\min f_5(x) = \left[\sum_{i=1}^{5} i\cos\left((i+1)x_1+i\right)\right]\times\left[\sum_{i=1}^{5} i\cos\left((i+1)x_2+i\right)\right]$	$x_1 \in [-10, 10]$ $x_2 \in [-10, 10]$	2	$f(x^*)=-186.7309$
$\min f_6(x) = \left(x_1^2+x_2^2\right)^{0.25}\times\left[\sin^2\left(50\left(x_1^2+x_2^2\right)^2\right)+1\right]$	$x_1 \in [-100, 100]$ $x_2 \in [-100, 100]$	2	$f(x^*)=0$
$\min f_7(x) = \left(x_1-10\right)^3+\left(x_2-20\right)^3$ s.t. $\begin{cases} -\left(x_1-5\right)^2-\left(x_2-5\right)^2+100\le 0 \\ \left(x_1-6\right)^2+\left(x_2-5\right)^2-82.81\le 0 \end{cases}$	$x_1 \in [-13, 100]$ $x_2 \in [0, 100]$	2	$f(x^*)=-6961.8139$
$\min f_8(x) = 0.5-\frac{\sin^2\left(\sqrt{x_1+x_2}\right)-0.5}{\left(1+0.001\left(x_1^2+x_2^2\right)\right)^2}$	$x_1 \in [-10, 10]$ $x_2 \in [-10, 10]$	2	$f(x^*)=-1$
$\min f_9(x) = x_1^2+x_2^2$ s.t. $\begin{cases} 2.5-x_1-x_2\ge 0 \\ x_2-x_1^2\ge 2 \end{cases}$	$x_1 \in [-100, 100]$ $x_2 \in [-100, 100]$	2	$f(x^*)=4$
$\min f_{10}(x) = \left(\frac{4}{3}x_1\right)^2+\left(-\frac{1}{3}x_1+x_2-2\right)^2+\left(-\frac{2}{3}x_1+x_2-1\right)^2+\left(-\frac{1}{3}x_1-1\right)^2$ s.t. $\begin{cases} \frac{2}{3}x_1+x_2+10\ge 0 \\ 10-\frac{2}{3}x_1-x_2\ge 0 \end{cases}$	$x_1 \in [-10, 10]$ $x_2 \in [-10, 10]$	2	$f(x^*)=1.4861$

V. CONCLUSION

In this paper, we propose a stage-specific gene expression framework (SGEF). The SGEF can embed into both single-objective optimization genetic algorithms and multiple-objective optimization genetic algorithms. Using multivariate analysis of variance, Welch-Satterthwaite's T-test, and non-parametric Wilcoxon Ranksum test, based on the experimental results of 17 benchmark and two practical optimization problems, we verified that the SGEF produces statistical significance improvements in the speed, efficiency, and quality of genetic algorithms. The leading cause of the outperformance of the SGEF can be stated as follows. Firstly, the combination of a two-tier coding scheme and a variational auto encoder could reduce the dimension of the search vector, and regularize the organization of the latent space. Secondly, a filter operator is designed according to the structure of the latent space, which maintains a trade-off balance between exploitation and exploration abilities. Thirdly, to speed up the calculation, operators can search for the optimal solution in a regularized latent space with a Gaussian distribution. Besides, this program is deployed in Python using the PyTorch library. The encoded mRNAs are stored as a tensor type, which can be processed using a GPU. Through this mechanism, we can reduce the waiting time between the IVAE training process and the genetic algorithm iteration process.

APPENDIX A
BASIC INFORMATION ABOUT SINGLE-OBJECT TEST FUNCTION

Basic information about single-object test function can be seen in Table 16.

APPENDIX B
PRESSURE VESSEL DESIGN PROBLEM

$$\begin{aligned}
&\min f(x) = 0.6224\, x_1x_3x_4 + 1.7781\, x_2x_3^2 \\
&\qquad\qquad + 3.1661\, x_1^2x_4 + 19.84\, x_1^2x_3 \\
&\text{Subject to: } g_1(x) = -x_1 + 0.0193\, x \\
&\qquad g_2(x) = -x_2 + 0.00954\, x_3 \le 0 \\
&\qquad g_3(x) = -\pi x_3^2x_4 - (4/3)\pi x_3^3 + 1,296,000 \le 0 \\
&\qquad g_4(x) = x_4 - 240 \le 0 \\
&\qquad 0 \le x_i \le 100, \quad i = 1, 2 \\
&\qquad 10 \le x_i \le 200, \quad i = 3, 4
\end{aligned}$$

APPENDIX C
TENSION-COMPRESSION SPRING DESIGN PROBLEM

$$\begin{aligned}
&\min f(\mathbf{x}) = (x_3 + 2)\, x_2x_1^2 \\
&\text{Subject to: } g_1(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}) = 1 - \frac{x_2^3x_3}{71785x_1^4} \le 0 \\
&\qquad g_2(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}) = \frac{4x_2^2 - x_1x_2}{12566\left(x_2x_1^3 - x_1^4\right)} \\
&\qquad + \frac{1}{5108x_1^2} - 1 \le 0 \\
&\qquad g_3(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}) = 1 - \frac{140.45\, x_1}{x_2^2x_3} \le 0 \\
&\qquad g_4(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}) = \frac{x_1 + x_2}{1.5} - 1 \le 0 \\
&\qquad 0.05 \le x_1 \le 2.0 \\
&\qquad 0.25 \le x_2 \le 1.3 \\
&\qquad 2.0 \le x_3 \le 15.0
\end{aligned}$$

REFERENCES

[1] C. Kim, R. Batra, L. Chen, H. Tran, and R. Ramprasad, "Polymer design using genetic algorithm and machine learning," *Comput. Mater. Sci.*, vol. 186, Jan. 2021, Art. no. 110067.
[2] Z. Soumaya, B. D. Taoufiq, N. Benayad, K. Yunus, and A. Abdelkrim, "The detection of Parkinson disease using the genetic algorithm and SVM classifier," *Appl. Acoust.*, vol. 171, Jan. 2021, Art. no. 107528.
[3] M. A. Elaziz, A. H. Elsheikh, D. Oliva, L. Abualigah, S. Lu, and A. A. Ewees, "Advanced metaheuristic techniques for mechanical design problems: Review," *Arch. Comput. Methods Eng.*, vol. 29, no. 1, pp. 695–716, Apr. 2021.
[4] A. Lashgari, H. Hosseinzadeh, M. Khalilzadeh, B. Milani, A. Ahmadisharaf, and S. Rashidi, "Transportation energy demand forecasting in Taiwan based on metaheuristic algorithms," *Energy Sources, A, Recovery, Utilization, Environ. Effects*, vol. 44, no. 2, pp. 2782–2800, Apr. 2022.
[5] A. Khan, Z. U. Rehman, M. A. Jaffar, J. Ullah, A. Din, A. Ali, and N. Ullah, "Color image segmentation using genetic algorithm with aggregation-based clustering validity index (CVI)," *Signal, Image Video Process.*, vol. 13, no. 5, pp. 833–841, 2019.
[6] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: Past, present, and future," *Multimedia Tools Appl.*, vol. 80, pp. 8091–8126, Oct. 2020.
[7] R. Shivgan and Z. Dong, "Energy-efficient drone coverage path planning using genetic algorithm," in *Proc. IEEE 21st Int. Conf. High Perform. Switching Routing (HPSR)*, May 2020, pp. 1–6.
[8] L. Abualigah and M. Alkhrabsheh, "Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing," *J. Supercomput.*, vol. 78, no. 1, pp. 740–765, May 2021.
[9] J. Huo, J. Liu, and H. Gao, "An NSGA-II algorithm with adaptive local search for a new double-row model solution to a multi-floor hospital facility layout problem," *Appl. Sci.*, vol. 11, no. 4, p. 1758, Feb. 2021.
[10] Q. Gu, X. Li, and S. Jiang, "Hybrid genetic grey wolf algorithm for large-scale global optimization," *Complexity*, vol. 2019, pp. 1–18, Feb. 2019.
[11] Y.-H. Jia, Y. Mei, and M. Zhang, "A two-stage swarm optimizer with local search for water distribution network optimization," *IEEE Trans. Cybern.*, vol. 53, no. 3, pp. 1667–1681, Mar. 2023.
[12] X. Wu, Y. Wang, J. Liu, and N. Fan, "A new hybrid algorithm for solving large scale global optimization problems," *IEEE Access*, vol. 7, pp. 103354–103364, 2019.
[13] S. M. Park, H. G. Yoon, D. B. Lee, J. W. Choi, H. Y. Kwon, and C. Won, "Optimization of physical quantities in the autoencoder latent space," *Sci. Rep.*, vol. 12, no. 1, May 2022.
[14] A. A. Ahmed, S. M. S. Darwish, and M. M. El-Sherbiny, "A novel automatic CNN architecture design approach based on genetic algorithm," in *Proc. Int. Conf. Advanced Intell. Syst. Inform.*, A. E. Hassanien, K. Shaalan, and M. F. Tolba, Eds. Cham, Switzerland: Springer, 2020, pp. 473–482.
[15] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing CNN architectures using the genetic algorithm for image classification," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3840–3854, Sep. 2020.
[16] J. Li, J. Zhang, J. Wang, Y. Zhu, M. J. Bah, G. Yang, and Y. Gan, "VAGA: Towards accurate and interpretable outlier detection based on variational auto-encoder and genetic algorithm for high-dimensional data," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2021, pp. 5956–5958.
[17] K. S. Tang, K. F. Man, Y. C. Ho, and S. Kwong, "A realizable architecture for genetic algorithm parallelism," *Control Eng. Pract.*, vol. 6, no. 7, pp. 897–903, Jul. 1998.
[18] Y. Liu, Q. Liao, J. Sun, M. Hu, L. Liu, and L. Zheng, "A heterogeneous parallel genetic algorithm based on SW26010 processors," in *Proc. IEEE 21st Int. Conf. High Perform. Comput. Commun., IEEE 17th Int. Conf. Smart City, IEEE 5th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Aug. 2019, pp. 54–61.

[19] J. R. Cheng and M. Gen, "Accelerating genetic algorithms with GPU computing: A selective overview," *Comput. Ind. Eng.*, vol. 128, pp. 514–525, Feb. 2019.
[20] G. Orphanides and D. Reinberg, "A unified theory of gene expression," *Cell*, vol. 108, no. 4, pp. 439–451, Feb. 2002.
[21] R. Yelin, D. Dahary, R. Sorek, E. Y. Levanon, O. Goldstein, A. Shoshan, A. Diber, S. Biton, Y. Tamir, R. Khosravi, S. Nemzer, E. Pinner, S. Walach, J. Bernstein, K. Savitsky, and G. Rotman, "Widespread occurrence of antisense transcription in the human genome," *Nature Biotechnol.*, vol. 21, no. 4, pp. 379–386, Mar. 2003.
[22] M. P. Deutscher, "E. Coli RNases: Making sense of alphabet soup," *Cell*, vol. 40, no. 4, pp. 731–732, Apr. 1985.
[23] D. Apirion, "Degradation of RNA in *Escherichia coli*," *Mol. Gen. Genet. MGG*, vol. 122, no. 4, pp. 313–322, Dec. 1973.
[24] S. Yokota, "Historical survey on chromatoid body research," *Acta Histochemica Et Cytochemica*, vol. 41, no. 4, pp. 65–82, 2008.
[25] C. Deneke, R. Lipowsky, and A. Valleriani, "Complex degradation processes lead to non-exponential decay patterns and age-dependent decay rates of messenger RNA," *PLoS ONE*, vol. 8, no. 2, Feb. 2013, Art. no. e55442.
[26] Y. Qin, H. K. Yalamanchili, J. Qin, B. Yan, and J. Wang, "The current status and challenges in computational analysis of genomic big data," *Big Data Res.*, vol. 2, no. 1, pp. 12–18, Mar. 2015.
[27] X.-B. Jin, W.-T. Gong, J.-L. Kong, Y.-T. Bai, and T.-L. Su, "PFVAE: A planar flow-based variational auto-encoder prediction model for time series data," *Mathematics*, vol. 10, no. 4, p. 610, Feb. 2022.
[28] P. Pennisi, L. Giallongo, G. Milintenda, and M. Cannarozzo, "Autism, autistic traits and creativity: A systematic review and meta-analysis," *Cognit. Process.*, vol. 22, no. 1, pp. 1–36, Oct. 2020.
[29] V. Picheny, T. Wagner, and D. Ginsbourger, "A benchmark of kriging-based infill criteria for noisy optimization," *Struct. Multidisciplinary Optim.*, vol. 48, no. 3, pp. 607–626, 2013.
[30] M. Molga and C. Smutnicki, "Test functions for optimization needs," *Test Functions Optim. Needs*, vol. 101, p. 48, Apr. 2005.
[31] W. Khatib and P. J. Fleming, "The stud GA: A mini revolution?" in *Parallel Problem Solving From Nature—PPSN V*, A. E. Eiben, T. Back, M. Schoenauer, and H.-P. Schwefel, Eds. Berlin, Germany: Springer, 1998, pp. 683–691.
[32] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 287–297, Nov. 1999.
[33] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 773–791, Oct. 2016.
[34] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.
[35] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.
[36] C.-C. Tsai, H.-C. Huang, and C.-K. Chan, "Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation," *IEEE Trans. Ind. Electron.*, vol. 58, no. 10, pp. 4813–4821, Oct. 2011.
[37] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 1, May 2002, pp. 825–830.
[38] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, "Modified distance calculation in generational distance and inverted generational distance," in *Proc. Int. Conf. Evol. Multi-Criterion Optim.* Cham, Switzerland: Springer, 2015, pp. 110–125.
[39] F. A. Hashim, K. Hussain, E. H. Houssein, M. S. Mabrouk, and W. Al-Atabany, "Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems," *Appl. Intell.*, vol. 51, pp. 1531–1551, Sep. 2020.
[40] Y. Song, F. Wang, and X. Chen, "An improved genetic algorithm for numerical function optimization," *Appl. Intell.*, vol. 49, no. 5, pp. 1880–1902, Dec. 2018.

PAN FENG received the B.S. degree in applied mathematics from the Minzu University of China, Beijing, China, in 1995, and the M.S. degree in computer software and theory program from Guizhou University, Guizhou, Guiyang, in 2008.

Since 2015, he has been a Professor with the School of Data Science and Information Engineering, Guizhou Minzu University. His research interests include machine learning, evolutionary computing, and systems science.

HAOZHOU SU received the B.S. degree in mathematics and applied mathematics from Beijing Forestry University, Beijing, China, in 2019. He is currently pursuing the M.S. degree in system science with Guizhou Minzu University, Guizhou, Guiyang.

His research interests include evolutionary computing and neural networks.