Xiao J, Shi ZH, Jiang JH *et al.* A locating method for reliability-critical gates with a parallel-structured genetic algorithm. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 34(5): 1136–1151 Sept. 2019. DOI 10.1007/s11390-019-1965-1

A Locating Method for Reliability-Critical Gates with a Parallel-Structured Genetic Algorithm

Jie Xiao[1], *Member*, *CCF*, Zhan-Hui Shi[1], *Member*, *CCF*, Jian-Hui Jiang[2,*], *Senior Member*, *CCF*, Xu-Hua Yang[1] Yu-Jiao Huang[1], and Hai-Gen Hu[1], *Member*, *CCF*

[1] *College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China*
[2] *School of Software Engineering, Tongji University, Shanghai 201804, China*

E-mail: xiaojiexqj@foxmail.com; shi_zhanhui@163.com; jhjiang@tongji.edu.cn
{xhyang, hyj0507, hghu}@zjut.edu.cn

Received January 11, 2019; revised June 3, 2019.

Abstract The reliability allowance of circuits tends to decrease with the increase of circuit integration and the application of new technology and materials, and the hardening strategy oriented toward gates is an effective technology for improving the circuit reliability of the current situations. Therefore, a parallel-structured genetic algorithm (GA), PGA, is proposed in this paper to locate reliability-critical gates to successfully perform targeted hardening. Firstly, we design a binary coding method for reliability-critical gates and build an ordered initial population consisting of dominant individuals to improve the quality of the initial population. Secondly, we construct an embedded parallel operation loop for directional crossover and directional mutation to compensate for the deficiency of the poor local search of the GA. Thirdly, for combination with a diversity protection strategy for the population, we design an elitism retention based selection method to boost the convergence speed and avoid being trapped by a local optimum. Finally, we present an ordered identification method oriented toward reliability-critical gates using a scoring mechanism to retain the potential optimal solutions in each round to improve the robustness of the proposed locating method. The simulation results on benchmark circuits show that the proposed method PGA is an efficient locating method for reliability-critical gates in terms of accuracy and convergence speed.

Keywords gate-level circuit reliability, locating reliability-critical gate, parallel-structured genetic algorithm, directing strategy, scoring mechanism

1 Introduction

Continually decreasing the size of the transistor in advanced very large scale integration (VLSI) circuits in recent years has led to lower power requirements and faster circuits[1−3]. However, shrinking the transistor size and lowering the supply voltage result in a significant reduction of the electrical node charge, which makes the circuits more prone to the reduction of their reliability allowance[4−6]. To avoid such disadvantages, hardening strategies, which require a locating method to identify the reliability-critical components for targeted hardening, are often used to improve circuit reliability[7]. Circuit reliability is defined as the probability of the correct functioning at the outputs of the circuit[8], and the reliability-critical components are those components that have a large impact on circuit reliability. In this paper, for any gate of a gate-level circuit, we define the criticality of gate reliability (CGR) as the probability that a gate is a reliability-critical gate.

The recent algorithms proposed for identifying reliability-critical components mainly include the graph theory based analysis method[9,10], the simulation-

Regular Paper

The work was supported by the National Natural Science Foundation of China under Grant Nos. 61972354, 61502422, 61432017, 61772199, 61773348, and 61503338, the Natural Science Foundation of Zhejiang Province of China under Grant Nos. LY18F020028, LY18F030023, LY18F030084, and LY17F030016, and the Innovative Experiment Project of Zhejiang University of Technology of China under Grant No. PX-68182112.

[*] Corresponding Author

based analysis method[1], and the evolutionary strategy based analysis method[11]. For example, using the reduced ordered binary decision diagrams, [9] counts the number of reliability-critical gates by input vectors, and [10] measures them by the reliability gradient with degree centrality. Locating and hardening the reliability-critical gates can improve circuit reliability at a low cost, but the methods in [1, 9] find it difficult to effectively deal with the contradiction between precision and complexity. The method in [10] is developed based on the average circuit reliability, which ignores the fact that different applied input vectors usually result in different circuit reliability, and some input vectors can cause the circuit reliability to be several orders of magnitude lower than the average circuit reliability[1,12]. The simulation-based analysis method is usually performed by simulating circuit behaviors to ensure accuracy. Although the credibility of its results has been unanimously recognized by the industry, it tends to have a large time consumption[13−15], such as the Monte Carlo (MC) method. The evolutionary strategy based analysis method identifies reliability-critical gates by running the evolutionary algorithms with specified rules in a specified area, which can often speed up the solution and find optimal solutions to the problems. Recently, the strategy has already attracted attention from the industry and has been successfully applied in some engineering fields[16,17], although building an efficient algorithm for the specified problems is a challenge. For example, by combining with a greedy hill climbing algorithm, [11] uses a recursive algorithm to locate the critical gates associated with the worst reliability input vectors. However, it is a huge challenge to accurately find the worst input vector in large-scale circuits. Critical gates are the gates that have direct impact on the reliability of the circuit's output signals, which is somewhat different from the reliability-critical gates defined in this paper. Further analysis determined that none of the methods mentioned above have presented the importance ranking of the reliability-critical gates except for the simulation-based analysis method applied in [1], which is not conducive to performing circuit reliability improvement with targeted hardening at a low cost.

Based on the above analysis and the existing computing power, we investigate further based on the evolutionary strategy to identify and rank the reliability-critical gates. Evolutionary algorithms based on the framework of genetic algorithms (GAs)[16,18,19], such as simulated annealing GA and quantum GA, have been developed and successfully applied to solve many optimization problems. For example, [11] finds that GA is more suitable for finding the worst circuit input vectors than the Hill Climbing and Simulated Annealing algorithms. And [16] presents a GA-based approach to select a suitable binary decision diagram ordering to optimize the objective parameters for circuit synthesis without performance degradation. Therefore, the GA-based framework is chosen in this study.

However, the CGR of a given gate depends not only on its position in the circuit but also on the circuit input vectors[4,6,20], which makes the analysis much more difficult. Much work remains to be done, including the targeted designs for the selection, crossover and mutation operations in the GA-based framework. Therefore, we develop the PGA method for this study. Firstly, we encode all gates with binary coding and generate the initial population using a specified rule. Secondly, we employ a reliable input vector oriented circuit reliability evaluation method with a high accuracy and a high speed as the fitness function for this proposed method. Then we construct the specified selection, crossover and mutation operations with direction functions. Thirdly, we construct an evaluation method for CGR. Finally, we extract the ordered reliability-critical gates and provide them to circuit designers for reference. Basically, the steps mentioned above boost the convergence speed by ensuring locating accuracy.

The rest of the paper is organized as follows. The related studies are introduced in Section 2, followed by an introduction of the PGA method for locating reliability-critical gates, including the evolutionary operations and the evaluation method for CGR in Section 3. The experiments and analysis are presented in Section 4, followed by conclusion remarks in Section 5.

2 Related Work

The related work is as follows. The circuit input vector oriented evaluation method for circuit reliability is described in Subsection 2.1. The simulation based analysis method for identifying the reliability-critical gates in a circuit is described in Subsection 2.2.

2.1 Sensitivity Calculation Algorithm

It was found that the input vector oriented evaluation method for circuit reliability needs several iterations in the calculation process, and its calculation speed and accuracy have a crucial influence on the effectiveness of the proposed method[4,11]. Therefore, an accurate and fast circuit reliability evaluation method is required. It is found that the sensitivity calculation algorithm (SCA) meets the requirements exactly and has a linear increase in time-space consumption with the number of gates[21]. Thus, the SCA is selected as the fitness calculation for the individuals in this study. The details are shown in Algorithm 1. The list of completed nodes (denoted as LC) represents the linked list in which the input information for any nodes can be obtained from the output information of their predecessor nodes.

Algorithm 1. SCA

```
Input: circuit netlist and applied input vector
Output: fitness of the input vector
1. Parse circuit netlist and initialize the related parameters
    1.1 Read the netlist, extract the information for the basic gates, build LC for the circuit, identify all primary inputs (PIs) and primary outputs (POs) of the circuit, and set $i = 1$.
    1.2 By the type, failure probability $pg$ and input number $ns$ of the basic gate, build probability transfer matrix (PTM) $\boldsymbol{PM}_{type\text{-}ns}$ and ideal transfer matrix (ITM) $\boldsymbol{IM}_{type\text{-}ns}$.
    1.3 Extract the input signals of PIs of the circuit and construct the corresponding primary input probability distribution $\boldsymbol{pipt}$, where $\boldsymbol{pipt} = [1, 0]$ when the input signal is 0; otherwise, $\boldsymbol{pipt} = [0, 1]$.
2. Calculate the faulty output probability distribution $\boldsymbol{fopt}_i$ and the ideal output probability distribution $\boldsymbol{iopt}_i$ for the $i$-th node in LC
    2.1 Extract the type, $pg_i$ and input number $mi$ for the $i$-th node in LC, generate its PTM $\boldsymbol{PM}_{type\text{-}nsi}$ and its ITM $\boldsymbol{IM}_{type\text{-}nsi}$; then, set $j = 1$.
    2.2 Extract the $j$-th input of $gi$: if it is the primary input of the circuit, assign $\boldsymbol{pipt}$ to the faulty input probability distribution $\boldsymbol{fipt}_{ij}$ and the ideal input probability distribution $\boldsymbol{iipt}_{ij}$; otherwise, extract $\boldsymbol{fipt}_{ij}$ and $\boldsymbol{iipt}_{ij}$ from $\boldsymbol{fopt}_q$ and $\boldsymbol{iopt}_q$ of the $q$-th node $(1 \leqslant q < i)$ in LC.
    2.3 If $j = m$, then go to step 2.4; otherwise, set $j = j + 1$ and go to step 2.2.
    2.4 Calculate $\boldsymbol{fipt}_i$ and $\boldsymbol{iipt}_i$ of $gi$ by
        $\boldsymbol{fipt}_i = \boldsymbol{fipt}_{i1} \otimes \boldsymbol{fipt}_{i2} \otimes \ldots \otimes \boldsymbol{fipt}_{imi}$,
        $\boldsymbol{iipt}_i = \boldsymbol{iipt}_{i1} \otimes \boldsymbol{iipt}_{i2} \otimes \ldots \otimes \boldsymbol{iipt}_{imi}$,
        where $\otimes$ denotes the operation of tensor product.
    2.5 Calculate $\boldsymbol{fopt}_i$ and $\boldsymbol{iopt}_i$ of $gi$ by
        $\boldsymbol{fopt}_i = \boldsymbol{fipt}_i \cdot \boldsymbol{PM}_{type\text{-}nsi}$,
        $\boldsymbol{iopt}_i = \boldsymbol{iipt}_i \cdot \boldsymbol{IM}_{type\text{-}nsi}$.
    2.6 If the output of $gi$ is the primary output of the circuit, then go to step 2.7; otherwise, go to step 2.8.
    2.7 Calculate the fitness $FR$ for the applied input vector
        $FR = 1 - \text{sum}(\boldsymbol{fopt}_i \cdot \boldsymbol{iopt}_i)$.
    2.8 If the end of LC is reached, then go to step 3; otherwise, set $i = i + 1$ and go to step 2.1.
3. Output $FR$
```

2.2 MC Method

It can be known from the analysis mentioned in the introduction that the credibility of the results obtained by the MC method has been widely recognized by the industry, which meets the requirement of the verification method in this study. Therefore, the MC method is used to verify the accuracy of the proposed method. The details are shown in Algorithm 2.

Algorithm 2. MC Method for Locating Reliability-Critical Gates

```
Input: circuit netlist
Output: ordered reliability-critical gates
1. Parse the circuit netlist and initialize the related parameters
    1.1 Read the circuit netlist and build the LC for the circuit.
    1.2 Extract the numbers of PIs and POs of the circuit, and denote them as $pm$ and $pn$, respectively.
    1.3 Initialize $pg$, and set the increment of $pg$ to $\Delta pg$, the experimental sample size to $N$, $j = 1$, $M = 0$, and $h = 1$.
2. Calculate the circuit reliability $Rc$
    2.1 Generate signals randomly for the primary inputs of the circuit, and set $k = 0$, $i = 0$, and $n = 0$.
    2.2 Set $i = i+1$, extract the $i$-th node from LC, and calculate its outputs for $pg = 0$ and $pg \neq 0$, respectively.
    2.3 If the primary output of the circuit is reached, set $k = k + 1$ and compare the results obtained from step 2.2. Then, go to step 2.4; otherwise, go to step 2.2.
    2.4 If the compared result is true, then $n_k = 1$; otherwise, $n_k = 0$.
    2.5 Set $n = n + n_k$.
    2.6 If the end of LC is reached, then go to step 2.7; otherwise, go to step 2.2.
    2.7 If $n = pn$, then set $M = M + 1$.
    2.8 If $j = N$, then output $Rc = M/N$; otherwise, set $j = j + 1$, and go to step 2.1.
3. Calculate the reliability increment of the $h$-th gate in the circuit and denote it as $IR_h$
    3.1 Extract the $h$-th node from LC, and set $pg = pg + \Delta pg$.
    3.2 Initialize $j = 1$ and $M = 0$, and then calculate the circuit reliability $Rg$ by the method presented in step 2.
    3.3 Set $IR_h = Rc - Rg$ and put it into the list of output nodes (denoted as LO).
    3.4 If the end of LC is reached, then go to step 4; otherwise, set $pg = pg - \Delta pg$ corresponding to the $h$-th node, set $h = h + 1$, and go to step 3.1.
4. Sort and output LO
```

3 PGA Framework Setup

As we all know, the GA-based framework includes the operations of selection, crossover, and mutation[22], and targeted designs have been used for different issues to provide effective results and boost the convergence speed[18,23,24]. In this study, to locate the reliability-critical gates in a circuit, some targeted designs for the PGA-based framework and the major operations involved in the evaluation are used to solve the problems encountered. Fig.1 depicts the configuration of the PGA-based framework, and Subsection 3.1 introduces the specified designs of the evolutionary operations. *Nge* and *NGE* denote the current generation number and the maximum number of generations, respectively.

The design of the PGA-based framework follows the principles of finding stable optimal solutions, preventing the possible occurrence of premature stagnation and boosting convergence speed. To achieve the aims, the following operations have been carefully designed. For example, the initial population consists of many well selected dominant individuals. A similarity analysis for each set of paired parents is performed. The parallel structured evolution operations for the proposed crossover (denoted as PRC) and the proposed mutation (denoted as PRM) are designed. The diversity analysis for the population is conducted. The preservation of the Hall-Group that retains the individuals with the best

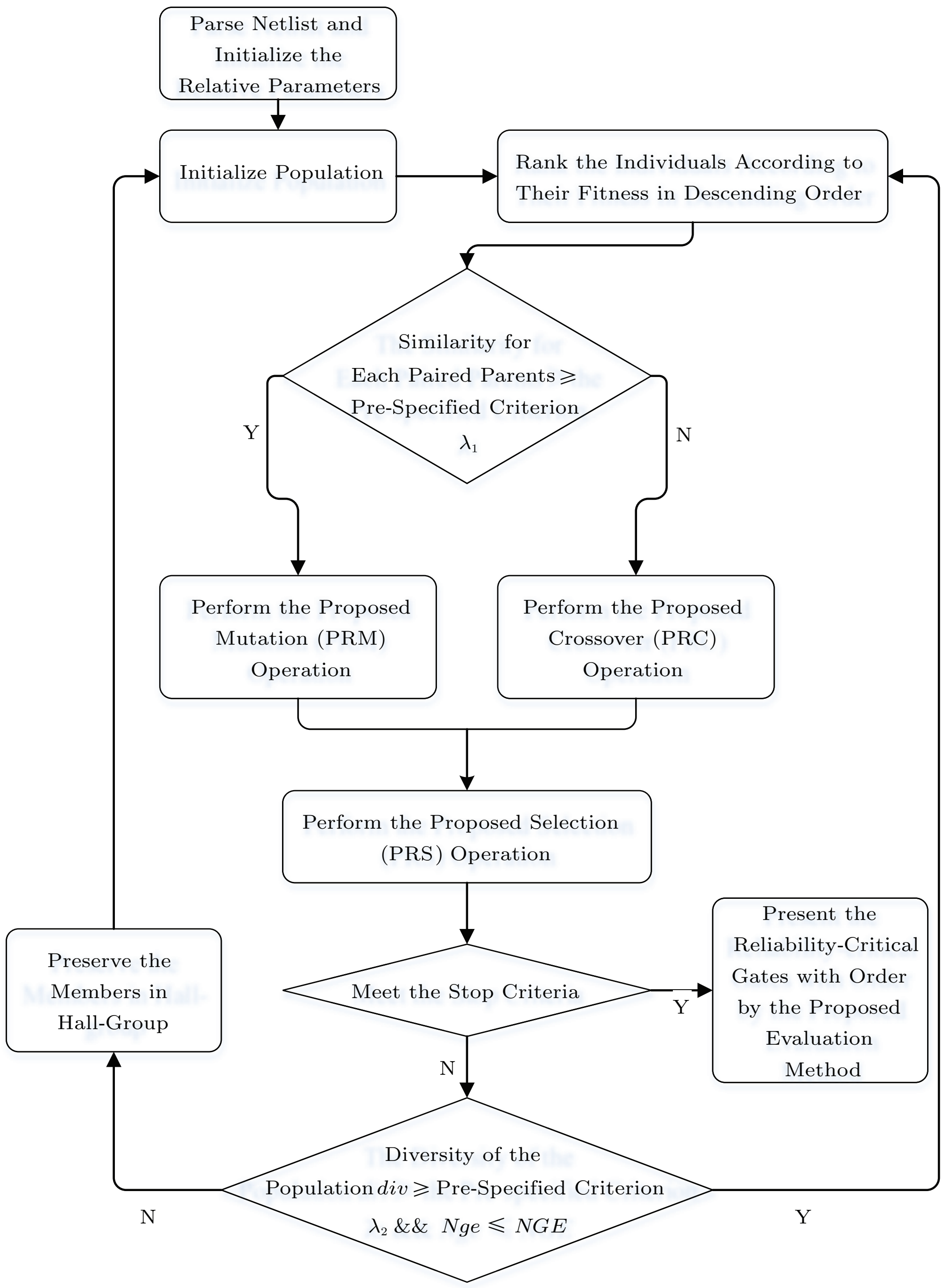
Parse Netlist and Initialize the Relative Parameters
Initialize Population
Rank the Individuals According to Their Fitness in Descending Order
Similarity for Each Paired Parents ⩾ Pre-Specified Criterion λ_1
Y
N
Perform the Proposed Mutation (PRM) Operation
Perform the Proposed Crossover (PRC) Operation
Perform the Proposed Selection (PRS) Operation
Preserve the Members in Hall-Group
Meet the Stop Criteria
Y
Present the Reliability-Critical Gates with Order by the Proposed Evaluation Method
N
Diversity of the Population $div \geqslant$ Pre-Specified Criterion λ_2 && $Nge \leqslant NGE$
N
Y

Fig.1. Flowchart of the PGA-based framework proposed in this paper.

fitness in each generation is designed, and the evaluation of the CGR is also designed. The details are as follows, where dominant individuals are the individuals with the best fitness in each calculation round.

3.1 Specified Designs of the Evolutionary Operations

The PGA-based framework mainly includes the operations of population initialization, selection, crossover, and mutation. The quality of their designs directly affects the performance of the algorithm for locating reliability-critical gates. Therefore, their targeted designs are as follows.

3.1.1 *Population Initialization*

To meet the requirements of the evaluation method for CGR presented in Subsection 3.2 and follow the principles of simplicity and ease of use, binary coding is used to encode gates. A gate represented by "1" is reliability-critical and its pg is set to $pg + \Delta pg$; otherwise, it is represented by "0" and its pg is set to pg. The encoding order of circuit gates is consistent with their order in LC.

To obtain the dominant initial population of reliability-critical gates, by drawing on the method presented in [25], we extract the gates from the results obtained from the multiple run simulations on the circuit, and the details are as follows. Firstly, we generate the individuals with a specified number nch in a random way, which ensures that the initial population is evenly distributed throughout the search space. Secondly, we select the individual with the best fitness as the member of the initial population by the tournament selection model, which improves the quality of the initial population. Thirdly, we repeat step 1 and step 2 until the size of the initial population is met. The process is shown in Fig.2, where nps is the size of the initial population, and the fitness for each individual is calculated by SCA.

3.1.2 *Similarity Analysis*

This operation analyzes the similarity for each set of paired parents denoted as sim (Fig.3 is an example of its calculation) to avoid unnecessary operations to boost the convergence speed. If the result is higher than the pre-specified criterion λ_1, it can be considered that the crossover operation offers a lower chance to explore better solutions in the current situation. Thus, the mutation operation is needed to improve diversity. Otherwise, the crossover operation can be run well in the current search space without creating new solutions using the mutation operation. Each set of paired parents consists of two adjacent individuals in this study. The design promotes the offspring inheriting superior genes from their parents in subsequent evolutionary operations, because the individuals in the population are ranked in descending order by their fitness. $N_{\rm LC}$ is the size of LC; $\boldsymbol{pat}_i$ and $\boldsymbol{pat}_{i+1}$ are the chosen paired parents and denote the i-th and the (i+1)-th parents, respectively; bitand and sum represent the bitwise AND and the summation operations, respectively.

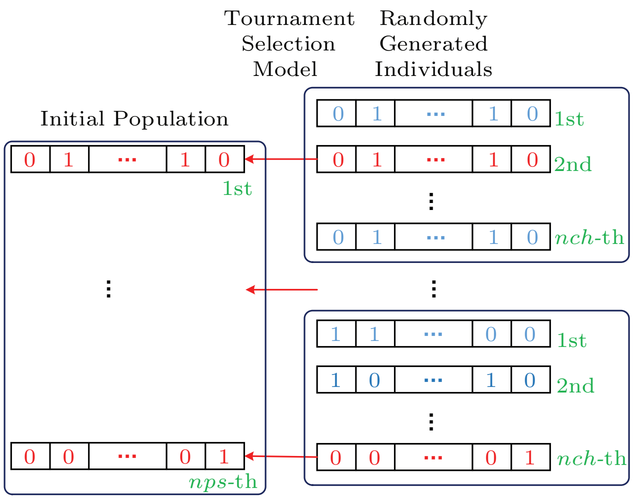

Fig.2. Process of population initialization.

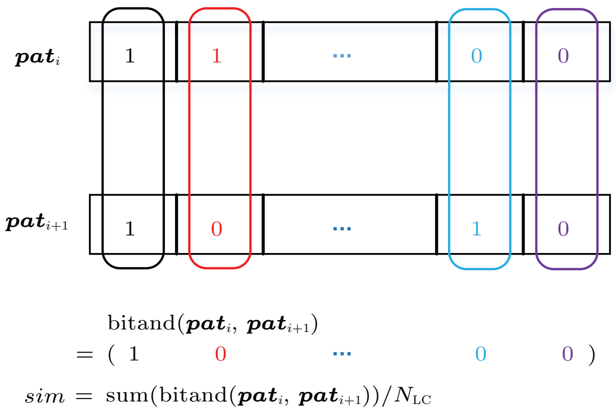
$\boldsymbol{pat}_i$
$\boldsymbol{pat}_{i+1}$
1 1 ··· 0 0
1 0 ··· 1 0
bitand($\boldsymbol{pat}_i$, $\boldsymbol{pat}_{i+1}$)
= (1 0 ··· 0 0)
sim = sum(bitand($\boldsymbol{pat}_i$, $\boldsymbol{pat}_{i+1}$))/$N_{\rm LC}$

Fig.3. Example of the similarity calculation for a set of paired parents.

3.1.3 *Crossover Operation*

To develop a self-adaptive crossover operation for locating circuit reliability-critical gates effectively, we require that each set of paired parents adopts the crossover probability matching their fitness in each generation. That is to say, we make the paired parents with approximate fitness have a lower crossover probability and others have a higher crossover probability. The operation is conducive to minimizing the unnecessary operation for crossover and offering a higher chance for exploring better solutions. Therefore, the one-point crossover operation with direction function is built. The details are presented in Algorithm 3.

```
Algorithm 3. PRC Operation
Input: $\boldsymbol{pt}_i$ and $\boldsymbol{pat}_{i+1}$
Output: $\boldsymbol{chd}_i$ and $\boldsymbol{chd}_{i+1}$
1. Generate an integer random number $rnc$ in $[1, N_{\rm LC} \times (2 - Nge/NGE)/2]$.
2. Generate the crossover probability by
   $pcr = pcr_{\min} + |fpat_i - fpat_{i+1}|/|fmax - fmin|$,
   where $pcr_{min}$ denotes the minimum crossover probability, and $fpat_i$, $fpat_{i+1}$, $fmax$ and $fmin$ denote the fitness of $\boldsymbol{pat}_i$, $\boldsymbol{pat}_{i+1}$, the first parent and the last parent in the current generation, respectively.
3. Generate a random number $rd$ in [0, 1].
4. If $rd < pcr$, perform the one-point crossover manipulation for the paired parents of $\boldsymbol{pat}_i$ and $\boldsymbol{pat}_{i+1}$ among their genes of 1 to $rnc$, and produce their candidate offspring of $\boldsymbol{chd}_i$ and $\boldsymbol{chd}_{i+1}$, the processes are depicted in Fig.4, then go to step 5; otherwise, go to step 6.
5. Perform the survivor selection of both the parents and their candidate offspring and preserve the two individuals with the best fitness as the current offspring.
6. End the algorithm.
```

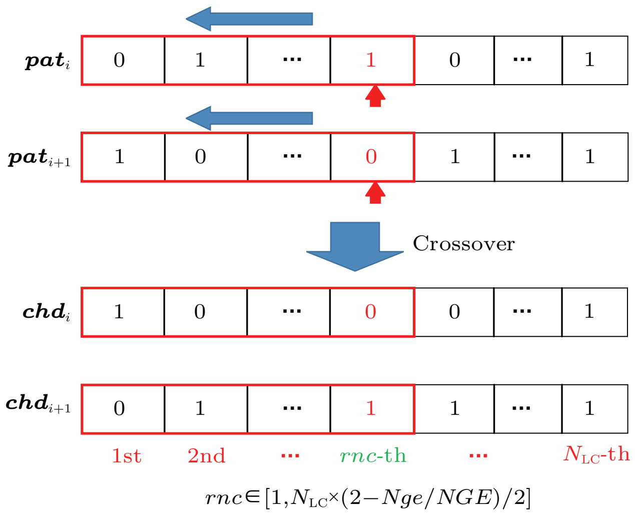

Fig.4. One-point crossover for $\boldsymbol{pat}_i$ and $\boldsymbol{pat}_{i+1}$.

According to the design of the PRC operation, the followings can be obtained. 1) The crossover bit rnc gradually moves forward with the increase of the evolutionary generation number Nge, which means that the exploration gradually turns from global to local. This result is based on the fact that the results gradually converge with the increase of Nge and the gates away from the circuit primary outputs commonly have less impact on circuit reliability[11]. 2) The relevant fitness information of the chosen paired parents is used for the adaptive control of the step size of crossover to increase the possibility of effective exploration, which helps protect the dominant individuals and prevent premature convergence. 3) The better individuals are selected to be the survivor offspring by the competition between the parents and their candidate offspring, which is an important manipulation for crossover direction.

3.1.4 *Mutation Operation*

The PRM operation is used to purposefully change the gene of an individual to improve population diversity, increase the probability of generating a better individual, and avoid being trapped by a local optimum. Each bit in the individuals denotes the state of a gate and the solutions tend to improve with the increase of Nge. We find that a self-adaptive mutation operation is needed and only small feasible perturbations are allowed to be performed on the population to improve the local search capability of the GA. Therefore, a single-point mutation operation is built in this study. The details are presented in Algorithm 4.

```
Algorithm 4. PRM Operation
Input: $\boldsymbol{pat}_i$
Output: $\boldsymbol{chd}_i$
1. Generate an integer random number $rnm$ in $[1, N_{\rm LC}]$ and take it as the mutation bit.
2. Generate the mutation probability by
   $pmt = pmt_{\min} + (1 - Nge/NGE)^b \times |fmax - fpat_i|$,
   where $b$ ($> 0$) is a parameter to control the decay rate of $pmt$, and $pmt_{min}$ denotes minimum mutation probability.
3. Generate a random number $rd$ in [0, 1].
4. If $rd < pmt$, perform single-point mutation for the $rnm$-th bit in $\boldsymbol{pat}_i$, which means that logical negation is performed on that bit, then go to step 5; otherwise, go to step 6.
5. Perform the survivor selection of both the parent $\boldsymbol{pat}_i$ and its candidate offspring $\boldsymbol{chd}_i$, and preserve the individual with the best fitness as the current offspring.
6. End the algorithm.
```

According to the description of the design of the PRM operation, the followings can be obtained. 1) The PRM operation offers a higher chance of population variations at the beginning of the evolution period, and the chance gradually decreases with the increase of Nge. 2) The operation can adaptively adjust pmt for each individual in the current generation through the relevant fitness information. The operation provides a higher mutation probability for poorer individuals and a lower mutation probability for better individuals. 3) The operation enables the direction function through the survivor selection of both the parent and its offspring, which is conducive to producing better individuals in each generation.

3.1.5 *Selection Operation*

To select the dominant individuals, some methods for selection operation have been presented[26,27], such as roulette wheel selection, tournament selection, rank-based selection, and the elite-preservation strategy. However, it was found that these methods are not suitable for selecting dominant individuals, mainly because they fail to preserve dominant individuals while ensuring population diversity for the issues encountered in this study.

To overcome the deficiency, combining with the advantages of the existing methods, we propose a new selection operation denoted as PRS, which is as follows. 1) The fitness of each individual facing different input vectors is determined, which helps truly reflect the fitness of each individual in the population. 2) The first *nso* individuals with the best fitness are taken as a part of the offsprings and the rest of the offsprings are generated randomly. The operation not only helps preserve dominant individuals, but also helps ensure population diversity. 3) The individual with the best fitness is put into the Hall-Group (HG) as a part of the candidate optimal solutions to the problem in this study, which avoids destruction in subsequent evolution operations. The details are shown in Algorithm 5, and Fig.5 is a relevant example.

```
Algorithm 5. PRS Operation
Input: a population (PPo), $pm$
Output: a new population (PPn) and an HG
1. Get the fitness of the individuals in PPo.
   1.1 Calculate the fitness of each individual for $niv$ input vectors generated in a random way,
       $niv = pm \times \beta$,
       where $\beta$ is a user-specified parameter.
   1.2 Extract the median of the results for each individual in step 1 and set the median as the fitness of the corresponding individual.
2. Sort the individuals by their fitness in descending order.
3. Create PPn
   3.1 Extract the first $nso$ individuals as a part of the offsprings and put them into PPn.
   3.2 Generate the rest of the offsprings in a random way and put them into PPn.
4. Put the individual with the best fitness into HG.
5. Output PPn and HG.
```

3.1.6 *Diversity Analysis*

To evaluate the diversity of the current population, true operations are used to prevent the possible occurrence of premature stagnation and make the proposed method more efficient at finding the optimal solution. In this paper, the standard deviation of the fitness for the current population (denoted as *div*, see in (1)) is used to identify population diversity. If $div < \lambda_2$, we refresh all individuals except the individuals in HG by regenerating a new population; otherwise, we perform the subsequent operations. Here, λ_2 is a pre-specified criterion, and f_i and $f_{\rm ave}$ denote the fitness of the i-th individual and the average fitness of all individuals in the current population, respectively.

$$div = \sqrt{\frac{1}{nps-1}\sum_{i=1}^{nps}(f_i - f_{\rm ave})^2}. \quad (1)$$

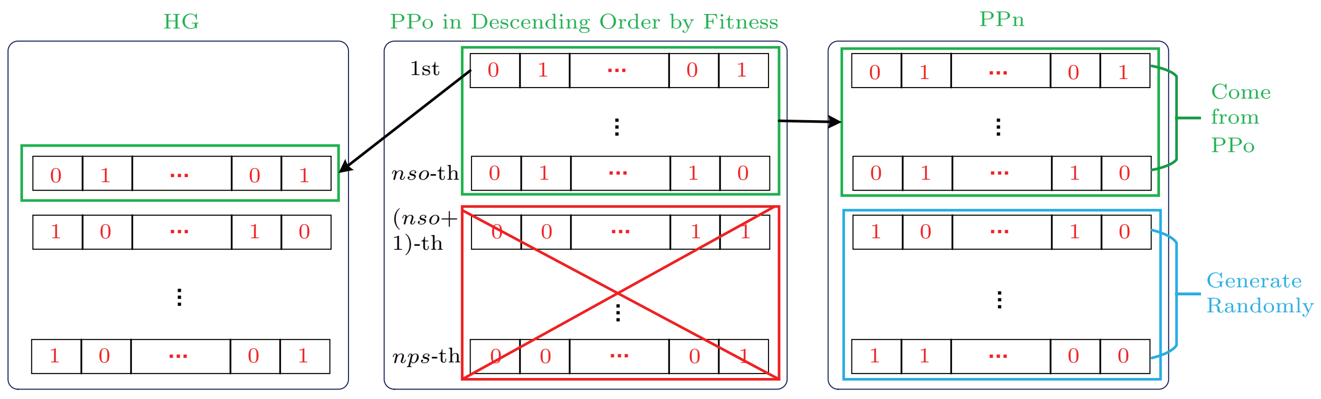

Fig.5. Example to illustrate the PRS.

3.2 Locating Algorithm of Reliability-Critical Gates

Evaluating the CGR of each gate in a circuit is an important prerequisite to ranking them. The dominant individuals commonly have a higher generated probability than other individuals throughout the calculation process. Therefore, a new method learned from the strategy of reliability statistics presented in the MC method[4,14], is proposed to calculate the CGR for a circuit. 1) We count the number of "1"s in each corresponding bit in all individuals in HG. 2) We calculate the frequencies of "1" in each bit in the previous step and set them as the corresponding CGR. 3) We rank the gates in descending order by their CGR. A relevant example of the calculation is shown in Fig.6, where gi is the i-th gate in LC, N_{gi} is the number of "1"s appearing in the bit corresponding to gi, $N_{\rm HG}$ is the size of HG, and RC_{gi} is the CGR of gi and $i = 1, 2, \dots, N_{\rm LC}$.

However, it was found that the result for the reliability-critical gates obtained by the statistics method is prone to instable when the CGR is approximate, which is inevitable in MC-based statistic methods unless a large number of simulation runs are performed[4,13,28]. Considering that we only care about the order of the CGR of gates in the circuit, strategies of running the above statistics method multiple times and reassigning CGR are adopted in this study. 1) The statistics method is performed Nsm times. 2) The CGR for each gate in each round is reassigned based on some specified rules. 3) The CGR of the Nsm rounds for each gate is calculated and the gates are sorted in descending order by their CGR. The details are shown in Fig.7 and Algorithm 6.

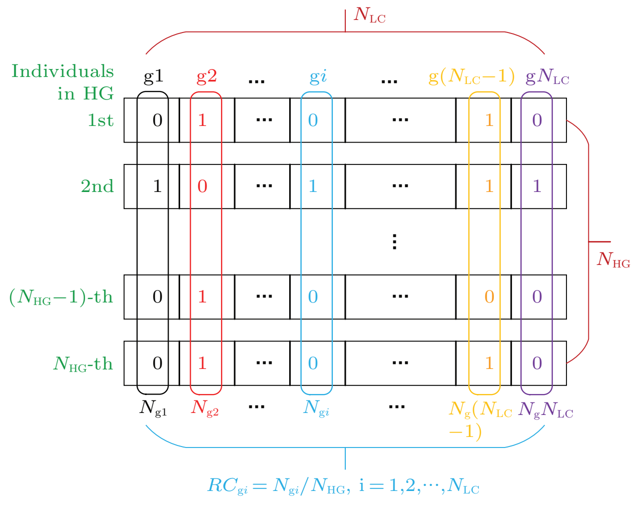

Fig.6. Example of the CGR evaluation for a circuit.

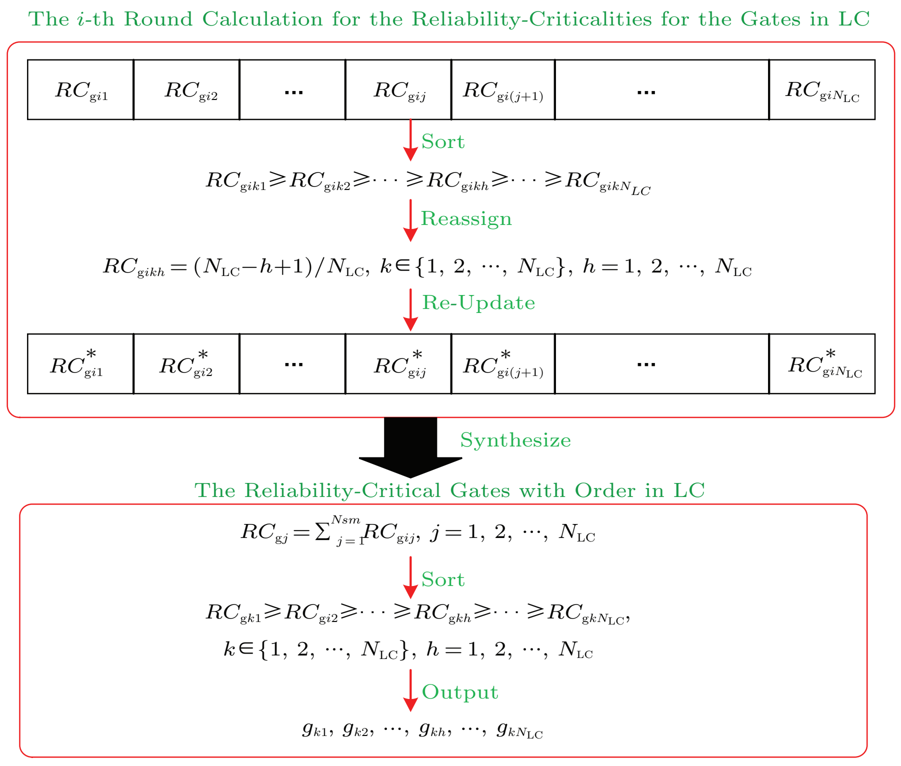
The i-th Round Calculation for the Reliability-Criticalities for the Gates in LC
RC_{gi1} RC_{gi2} ⋯ RC_{gij} $RC_{gi(j+1)}$ ⋯ $RC_{giN_{\rm LC}}$
Sort
$RC_{gik1} \geqslant RC_{gik2} \geqslant \cdots \geqslant RC_{gikh} \geqslant \cdots \geqslant RC_{gikN_{LC}}$
Reassign
$RC_{gikh} = (N_{\rm LC} - h + 1)/N_{\rm LC}$, $k \in \{1, 2, \cdots, N_{\rm LC}\}$, $h = 1, 2, \cdots, N_{\rm LC}$
Re-Update
RC^*_{gi1} RC^*_{gi2} ⋯ RC^*_{gij} $RC^*_{gi(j+1)}$ ⋯ $RC^*_{giN_{\rm LC}}$
Synthesize
The Reliability-Critical Gates with Order in LC
$RC_{gj} = \sum_{j=1}^{Nsm} RC_{gij}$, $j = 1, 2, \cdots, N_{\rm LC}$
Sort
$RC_{gk1} \geqslant RC_{gi2} \geqslant \cdots \geqslant RC_{gkh} \geqslant \cdots \geqslant RC_{gkN_{\rm LC}}$,
$k \in \{1, 2, \cdots, N_{\rm LC}\}$, $h = 1, 2, \cdots, N_{\rm LC}$
Output
$g_{k1}, g_{k2}, \cdots, g_{kh}, \cdots, g_{kN_{\rm LC}}$

Fig.7. Improved method for calculating RC_{gj}, $j \in \{1, 2, \dots, Ng\}$.

```
Algorithm 6. CGR Calculation for Gates in a Circuit
Input: individuals in HGs
Output: reliability-critical gates with order
1. Initialize $i = 1$, $N_{\rm HG}$ and $Ng = N_{\rm LC}$
2. If $i \leqslant Nsm$, then go to step 3; otherwise, go to step 4.
3. Perform the MC-based statistics method for the $i$-th round and obtain the CGR.
    3.1 Count the number of "1"s appearing in the $j$-th bit in all individuals in the $i$-th HG$_i$ ($j = 1, 2, \ldots, N_{\rm LC}$).
    3.2 Calculate the CGR for the $j$-th gate in LC corresponding to the $j$-th bit in individuals using the results obtained from step 3.1 ($j = 1, 2, \ldots, N_{\rm LC}$).
    3.3 Sort the obtained CGR in descending order.
    3.4 Get the first $Ng$ results obtained from step 3.3, reassign them by $RC_{{\rm g}ikh} = (Ng-h+1)/Ng$ ($k \in \{1, 2, \ldots, Ng\}$, $h = 1, 2, \ldots, Ng$). $RC_{{\rm g}ikh}$ is the CGR with the $h$-th order in the $i$-th round calculation, which corresponds to the $k$-th gate in LC.
    3.5 Perform $i = i + 1$ and go to step 2.
4. Aggregate the CGR of the $j$-th gate in LC by
       $RC_{{\rm g}j} = \sum_{i=1}^{Nsm} RC_{{\rm g}ij}$, $j = 1, 2, \ldots, Ng$.
5. Sort the results obtained from step 4 in descending order. If different gates encounter the same CGR, deal with them using the rules presented in Subsection 3.2.
6. Output the gates in descending order of their CGR.
```

Sometimes, different gates encounter the same CGR. We deal with this situation by the following two rules, for which the relevant example is shown in Fig.8. Let $D_{{\rm g}k}$ denote the distance of ${\rm g}_k$ to the primary output of the circuit and $RC_{{\rm g}ij}$ be the CGR of ${\rm g}_j$ in the i-th round calculation. Supposing $RC_{{\rm g}(k-1)} = RC_{{\rm g}k} = RC_{{\rm g}(k+1)}$, and $D_{{\rm g}(k-1)} > D_{{\rm g}k} = D_{{\rm g}(k+1)}$, according to the following two rules, the gates in descending order of their CGR are: ${\rm g}_{k+1} > {\rm g}_k > {\rm g}_{k-1}$.

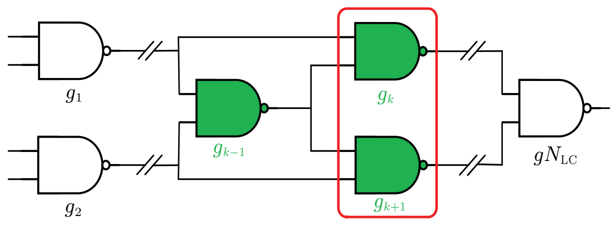

Fig.8. Example to illustrate the rules of ranking the gates with the same CGR.

(1) Gates closer to the primary output of the circuit are considered to have higher CGR than the others.

(2) Gates with a larger sequence number in LC are deemed to have higher CGR than the others when the gates with the same CGR have the same distance to the primary output of the circuit.

It should be noted that only the CGR belonging to the same basic gate allows us to perform summation operation. The distance of the specified gate to the primary output of the circuit is defined as the minimum number of gates on the corresponding paths.

By performing the reassignment strategy for the CGR and the synthesis of the results for multiple runs, the method presented in Algorithm 6 effectively alleviates the less stable problem of the results obtained by the MC-based statistics method and deals well with the situation that has the same CGR for different gates.

Using population initialization, the operations of selection, crossover, and mutation, and the evaluation method for CGR given above, a locating method for reliability-critical gates, as shown in Algorithm 7, is proposed based on the PGA-based framework presented in Fig.1.

In summary, to effectively locate the reliability-critical gates in a circuit, we design a parallel structured GA with direction function according to circuit reliability, which consists of the following seven major functions. The first major function is population initialization, which selects the dominant individuals oriented to the different input vectors as the members of the initial population, thus improving the quality of the initial population. The second function is the parallel-structured framework for GA, which organizes the operations of PRC and PRM according to the similarity of the chosen paired parents and helps avoid unnecessary operations. The third one is the PRC operation, which can adaptively adjust the search space and the crossover probability by the evolutionary generation number and the fitness information of the chosen paired parents. In addition, the operation can preserve the dominant individuals for the offspring by performing the competition strategy between the parents and their offspring. The fourth one is the PRM operation, which can dynamically and adaptively adjust the mutation bit in each generation and the mutation probability using the evolutionary generation number and the fitness information of the chosen individual. The operation preserves the dominant individual for the offspring using survivor selection. The operations of PRC and PRM improve the local search capability of the GA and prevent the population from prematurely converging to a sub-optimal solution. The fifth one is the PRS operation, which preserves a specified number of the best individuals for the new population, randomly generates the rest of the individuals for the new population, and chooses the best individual in each generation as the potential optimal solution. The operation helps boost the convergence speed of the GA while ensuring population diversity. The sixth one is the diversity analysis for the population in each generation, which helps prevent premature stagnation. The seventh one is the CGR calculation for gates, which scores each gate in LC to identify and sort the reliability-critical gates according to their CGR. This operation helps improve the robustness of the proposed algorithm.

Algorithm 7. Locating Reliability-Critical Gates in a Circuit

```
Input: sub-circuit netlist
Output: ordered reliability-critical gates
1. Parse the sub-circuit netlist and initialize the related parameters.
   1.1 Read the sub-circuit netlist and build an LC for the sub-circuit with only one primary output.
   1.2 Extract the primary inputs and basic gates from the sub-circuit, encode the basic gates with binary coding, and then initialize $i = 1$, $Nsm$, $NGE$.
2. Build an initial population pop-init for the reliability-critical gates by the method presented in Subsection 3.1, and set $j = 1$.
3. Create a new $HG_i$ and put the best individual in each generation into $HG_i$.
   3.1 Sort the individuals in the current population in descending order according to their fitness and set $k = 1$.
   3.2 Calculate the similarity of the $k$-th and the $(k+1)$-th paired parents (denoted as sim). If $sim < \lambda_1$, then go to step 3.3; otherwise, go to step 3.4.
   3.3 Perform the PRC operation for the paired parents using the method presented in Subsection 3.1.3, and then go to step 3.5.
   3.4 Perform the PRM operation for the $k$-th and the $(k+1)$-th individuals using the method presented in Subsection 3.1.4, and then go to step 3.5.
   3.5 If $k > N_{\mathrm{LC}}$, then go to step 3.6; otherwise, set $k = k + 2$ and go to step 3.2.
   3.6 Perform the PRS operation for the new population generated by the operations of PRC and PRM, then renew $HG_i$ by the PRS operation.
4. If $i > Nsm$, then go to step 7; otherwise, go to step 5.
5. Calculate div for the population obtained from step 3 using the method presented in Subsection 3.1.5. If $div \geqslant \lambda_2$ and $j \leqslant NGE$, then set $j = j + 1$ and go to step 3.1; otherwise, calculate $RC_{gih}$ for the $h$-th gates in LC using the reassignment strategy presented in Algorithm 6 ($h = 1, 2, \ldots, N_{\mathrm{LC}}$).
6. Set $i = i + 1$ and go to step 2.
7. Calculate $RC_{gh}$ for the $h$-th gate in LC by
       $RC_{gh} = \sum_{i=1}^{Nsm} RC_{gih}$, $h = 1, 2, \ldots, N_{\mathrm{LC}}$,
   and output the reliability-critical gates in descending order with respect to their CGR.
```

4 Simulation Experiment

We now demonstrate the accuracy and efficiency of the proposed method PGA through simulations performed on some typical circuits using a Dell T7910 computer with a 2.1 GHz processor and 16 GB RAM. For better comparison and analysis, the results are analyzed in three parts. We firstly demonstrate the accuracy of this method using the MC method presented in Algorithm 2, a stochastic method and the method presented in [11]. We then analyze the efficiency of the operations presented in this method through the comparison with some traditional operations. We then show some application examples. When all primary input signals are in the ideal state and are uniformly distributed, $nps = 20$, $\Delta pg = 0.05$, $pg = 0.05$, $Nsm = 10$, $pcr_{\min} = 0.4$, $pmt_{\min} = 0.01$, $\lambda_1 = 0.8$, $\lambda_2 = 0.01$, and $NGE = 200$ unless otherwise stated, and each gate is represented by their output wire marking.

4.1 Accuracy

Owing to the huge time consumption of the MC method and the convenience of displaying results, some small circuits, such as C17 and 74-series sub-circuits, are simulated by the MC method. Some large circuits, such as some ISCAS85 benchmark sub-circuits, simulated by the stochastic method, are used to test the accuracy of the proposed method. The results are listed in Table 1 and Table 2, where the reliability-critical gates in <> are sorted in descending order with respect to their CGR, while the gates outside <> are unordered. The primary inputs are generated randomly, the parameters $nch = 100$, $b = 4$, $\beta = 10$, and $N = 10^5$ are set empirically[29]. The circuit name-number refers to the sub-circuit whose primary output is the specified number. For example, C17-22 represents the sub-circuit of C17 with 22 POs; only 10 gates are presented in Table 1 and Table 2 for circuits with more than 10 gates. The stochastic method is performed as follows. 1) A specified number of reliability-critical gates are generated randomly. 2) The fitness of the reliability-critical gates using SCA is calculated. 3) The above steps are repeated until the population size requirement is met. 4) The reliability-critical gates with the best fitness are taken as the optimal solution to the problem in this study.

It can be seen from the comparison results in Table 1 and Table 2 that the order of the reliability-critical gates provided by the proposed method is approximate to that of the MC method on the small circuits, although the order of some gates is reversed, such as gates 72 and 73 in circuit 74148-79, and gates 32 and 38 in circuit 74283-101. The results obtained by [11] have a certain degree of accuracy loss, especially on larger circuits, such as circuit 74283-101 and circuit C5315-4737. On average, the proposed method is

Table 1. Comparisons of Identified Reliability-Critical Gates and Time Consumption of Different Methods on Small Circuits

Method	Circuit Name	Identified Reliability-Critical Gates	Run Time (s)
PGA	C17-22	<22, 16, 10, 11>	50.62
	74182-63	<63, 62, 61, 60, 59, 49>	151.35
	74157-40	<40, 36, 32, 22, 11, 17>	75.21
	74148-79	<79, 70, 71, 72, 73, 52, 32, 35, 24, 11>	4 771.92
	74283-101	<98, 86, 101, 38, 32, 78, 49, 88, 89, 87>	6 912.87
MC	C17-22	<22, 16, 10, 11>	2 476.90
	74182-63	<63, 62, 61, 60, 59, 49>	9 298.12
	74157-40	<40, 36, 32, 22, 11, 17>	4 528.23
	74148-79	<79, 70, 71, 73, 72, 52, 32, 35, 24, 11>	45 630.32
	74283-101	<98, 86, 101, 32, 78, 38, 49, 89, 88, 87>	58 040.21
Method in [11]	C17-22	22, 16, 11, 10	27.33
	74182-63	63, 62, 61, 60, 59, 49	106.21
	74157-40	40, 36, 22, 17, 11	62.90
	74148-79	79, 73, 72, 71, 70, 52, 35	347.70
	74283-101	101, 98, 89, 88, 87, 86, 78, 72, 61, 49	1 392.31

Table 2. Comparison of Identified Reliability-Critical Gates of Different Methods on Large Circuits

Method	Circuit Name	Identified Reliability-Critical Gates	Fitness
PGA	C432-223	<199, 223, 130, 134, 168, 180, 126, 122, 174, 150>	0.185 1
	C499-753	<655, 721, 753, 275, 367, 261, 258, 262, 305, 354>	0.218 8
	C880-767	<588, 552, 501, 409, 499, 498, 360, 366, 500, 533>	0.492 6
	C2670-2387	<1459, 2126, 1831, 1066, 1458, 1068, 2266, 1465, 1457, 1606>	0.376 2
	C3540-3195	<1253, 1254, 1258, 1279, 1328, 1508, 798, 706, 740, 1257>	0.541 4
	C5315-4737	<4273, 4737, 2891, 1475, 3615, 1482, 3691, 1216, 2882, 2877>	0.198 8
	C6288-1901	<1825, 549, 1824, 594, 1374, 639, 1373, 1404, 1582, 1687>	0.432 5
	C7552-10112	<3343, 6957, 1194, 2935, 9288, 2331, 1938, 762, 2674, 10035>	0.471 0
Stochastic method	C432-223	118, 122, 134, 142, 146, 150, 154, 168, 199, 223	0.154 6
	C499-753	250, 261, 277, 290, 340, 344, 393, 600, 655, 721	0.182 8
	C880-767	301, 304, 307, 363, 407, 426, 498, 550, 552, 588	0.490 4
	C2670-2387	509, 1042, 1066, 1457, 1458, 1459, 1463, 1464, 1465, 1606	0.343 9
	C3540-3195	782, 1252, 1255, 1256, 1520, 1721, 1756, 2181, 2379, 2746	0.528 5
	C5315-4737	1216, 1475, 1482, 2891, 3614, 3691, 4032, 4033, 4273, 4737	0.138 5
	C6288-1901	591, 1371, 1372, 1373, 1401, 1450, 1508, 1686, 1824, 1825	0.402 2
	C7552-10112	1194, 1983, 2344, 4769, 5959, 6547, 8326, 8604, 9904, 10035	0.468 3
Method in [11]	C432-223	223, 199, 180, 177, 174, 171, 168, 165, 162, 159	0.173 3
	C499-753	753, 721, 655, 607, 600, 593, 592, 582, 581, 580	0.204 5
	C880-767	660, 588, 551, 550, 533, 530, 501, 500, 499, 498	0.491 6
	C2670-2387	2266, 2126, 1831, 1615, 1465, 1464, 1463, 1068, 1053, 1042	0.423 7
	C3540-3195	3195, 2962, 2746, 2745, 2376, 2180, 1756, 1520, 1263, 686	0.485 8
	C5315-4737	4737, 4273, 4032, 3691, 2891, 2882, 1475	0.170 0
	C6288-1901	1901, 1825, 1824, 1763, 1717, 1687, 1628, 1582, 1512, 1511	0.423 0
	C7552-10112	10112, 10035, 9904, 9903, 9635, 9288, 9252, 8326, 7744, 6968	0.460 3

approximately 10 times faster than the MC method, and the method presented in [11] is approximately 4.5 times faster than the proposed method. The proposed method can identify the reliability-critical gates in descending order in the specified generations, while the stochastic method and the method presented in [11] cannot. The fitness obtained by the proposed method is better than that by the stochastic method and the method in [11] on the large circuits, and the order of the identified reliability-critical gates obtained by the proposed method is efficient, which has been verified by the following tests. The fitness obtained by the method in [11] is generally better than that by the stochastic method on the large circuits except for circuit C3540-

3195 and circuit C7552-10112. Furthermore, further tests show that the orders of gates 72 and 73, 32 and 38 are $<73, 72>$ and $<32, 38>$, respectively. In other words, the MC method has a higher accuracy than the proposed method for the given circuits.

The reasons are as follows. The MC method ensures its accuracy via large-scale simulations, which tend to require a lot of time. The proposed method ensures its accuracy using self-adaptive evolution for individuals and protection for dominant individuals, which boosts the convergence speed of the calculations and improve the probability of finding optimal solutions. The stochastic method obtains a solution from huge individuals generated randomly, which tends to consume a large amount of time and has a small probability of producing an optimal solution. The method in [11] identified the critical gates associated with the worst input vector. However, finding the worst input vector is a huge challenge, including time consumption and calculation accuracy, and the worst input vector cannot be used to effectively represent the entire space of input vectors. Also, the operation without ranking the critical gates is conducive to reducing the computational complexity. Moreover, the size of the applied input vectors is also one of the crucial factors affecting the effectiveness of the above methods, including locating accuracy and computing complexity. For example, the use of more input vectors can improve the accuracy of the above methods, but it also leads to higher time consumption[6].

In fact, it can be inferred from Table 1 and Table 2 that as the size of circuit increases, the search space increases dramatically, which often causes a huge computational cost to ensure locating accuracy and tends to reduce the practicability of the proposed method. To extend the scalability of the proposed method, a feasible locating solution to large-scale circuits could be a combination of modular processing of a circuit with parallel computing of modules, but left for future research.

4.2 Performance

To analyze the performance (such as NGC and fitness) of the proposed method, circuit 74283-101 and circuit C2670-2387 are randomly selected from Table 1 and Table 2 to perform the following comparative tests respectively.

Test-1. The initial population is randomly generated under the condition that other operations remain consistent with the proposed method.

Test-2. The traditional operations of crossover and mutation[30] are adopted under the condition that other operations remain consistent with the proposed method. The traditional crossover and mutation operations are denoted as *Test*-2c and *Test*-2m, respectively, and $pcr = 0.9$ and $pmt = 0.001$ for the traditional method[30].

Test-3. The roulette wheel selection operation[30] is adopted under the condition that other operations remain consistent with the proposed method.

Test-4. Diversity analysis is removed from the proposed method with other conditions remaining the same. The corresponding results are shown in Table 3, where NGC is the number of generations required for convergence of the experiments, and "–" means that the test cannot converge in the given generations. The process of the convergence of *Test*-4 is shown in Fig.9.

Test-5. Serial-structured GA is performed under the condition that other operations remain consistent with the proposed method, and the results are shown in Table 4.

Table 3. Performance Comparisons of the Proposed Method with *Test*-1, *Test*-2, *Test*-3 and *Test*-4

	74283-101		C2670-2387	
	NGC	Fitness	NGC	Fitness
Proposed method	124	0.466 6	9	0.457 3
Test-1	79	0.456 9	180	0.457 3
Test-2c	–	–	–	–
Test-2m	130	0.461 7	52	0.457 3
Test-3	–	–	–	–
Test-4	144	0.456 7	56	0.457 3

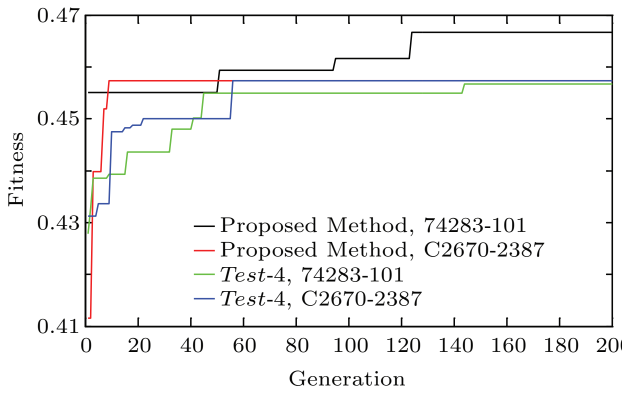

Fig.9. Performance comparisons of the proposed method with *Test*-4.

The convergence speed of the proposed method is faster than that by other tests on the above mentioned circuits. For example, the proposed method reaches convergence when its generation is 9, while *Test*-1 does when its generation is 180 on C2670-2387. The proposed method, *Test*-1, *Test*-2m, *Test*-4 and *Test*-5 can reach convergence when *NGE* is in the range from 1 to 200, while other tests cannot. The fitness obtained by the proposed method is the same as or better than that by other tests. For example, the fitness obtained by the proposed method is better than that by *Test*-1 on the circuit 74283-101, while the fitness obtained by this method is the same as that by *Test*-1 on the circuit C2670-2387. The effect of both *Test*-2c and *Test*-3 on the fitness of the circuits is higher than that by *Test*-1, *Test*-2m, *Test*-4 and *Test*-5. Finally, the structure applied in the proposed method is superior to *Test*-5. For example, the proposed method can converge faster than *Test*-5 on the applied circuits, and the fitness obtained by the proposed method is approximately equal to or better than that by *Test*-5 on the test circuits. For example, the fitness of circuit C432-223 obtained by the proposed method and *Test*-5 is 0.185 1 and 0.184 9, respectively.

Table 4. Performance Comparisons of Proposed Method with *Test*-5

Circuit	Proposed Method		*Test*-5	
	NGC	Fitness	NGC	Fitness
C432-223	78	0.185 1	144	0.184 9
C499-753	9	0.218 8	189	0.217 1
C880-767	57	0.492 6	70	0.483 7
C2670-2387	9	0.457 3	38	0.456 2
C3540-3195	52	0.541 4	73	0.535 0
C5315-4737	115	0.198 8	181	0.194 9
C6288-1901	69	0.432 5	164	0.423 4
C7552-10112	126	0.471 0	148	0.470 2

The dominant initial population consists of individuals with good fitness, which reduces the search space of the algorithm and boosts its convergence speed. The traditional operations of crossover and mutation are always performed with fixed *pcr* and *pmu*, and they do not preserve the generated dominant individuals. The operations require larger NGC to meet convergence, especially for the crossover operation. The major reasons are that the chance of performing crossover is much higher than that of performing mutation in this study and the fixed *pcr* is much higher than the fixed *pmu*. The roulette-based selection operation selects the individuals for subsequent operations randomly, which helps prevent the search from being trapped by a local optimum but increases the convergence cost. The diversity analysis not only helps avoid unnecessary calculations, which boosts the convergence speed of the algorithm, but also helps prevent premature convergence. The serial-structured GA performs the operations of crossover and mutation on all individuals, rather than adopts different coping strategies adapted to the fitness of the individuals, which tends to damage dominant individuals and slow the convergence speed. The proposed method avoids the deficiencies presented in the tests of *Test*-1 to *Test*-5, which results in fast convergence speed and good searching performance on the circuits. In short, it can be inferred from the above analysis that the proposed method is superior to the traditional GA method in both locating accuracy and convergence speed. Further analysis finds that better results, in terms of both accuracy and convergence speed, are possible with further customization of the empirical parameters (such as $pcr_{\min}$, $pmt_{\min}$, λ_1 and λ_2), but we leave it for future research.

4.3 Application

To introduce the practicability of the proposed method, the following four hardening schemes are performed for the selected gates.

Scheme *A*: hardening the first reliability-critical gate presented in Table 2.

Scheme *B*: hardening the first five reliability-critical gates presented in Table 2.

Scheme *C*: hardening one gate randomly selected from the circuits presented in Table 2.

Scheme *D*: hardening five gates randomly selected from the circuits presented in Table 2.

Then, the reliability increments *R*s are compared before and after the hardening schemes are performed. The results are shown in Table 5.

It can be seen from the comparison results in Table 5 that the effects of scheme *A* and scheme *D* on circuit reliability are superior to those of scheme *C*. The effects of scheme *B* on circuit reliability are better than those of scheme *A*, scheme *C* and scheme *D*. Finally, scheme *A* is superior to scheme *D* on some circuits, while scheme *D* is superior to scheme *A* on other circuits. For example, ΔRs obtained from scheme *A* are higher than those from scheme *D* on circuits C499-753 and C5315-4737, while ΔRs obtained from scheme *D* are higher than those from scheme *A* on the circuits C2670-2387 and C7552-1011.

Scheme *A* and scheme *B* harden the selected gates that have a considerable effect on circuit reliability, while scheme *C* and scheme *D* harden the gates randomly selected. The former schemes produce definite results and improve circuit reliability at a low cost at early stages of circuit design. Scheme *C* and scheme *D* generate indefinite and unpredictable results, which makes them hard to optimize the trade-off between reliability and area-power-delay. Moreover, although hardening more gates can further improve circuit reliability, it also leads to higher cost in other areas. Therefore, hardening the gates that have a greater effect on circuit reliability is a good method for improving circuit reliability at a low cost.

Table 5. ΔRs for Circuits in Different Hardening Schemes

Circuit	ΔRs			
	Scheme *A*	Scheme *B*	Scheme *C*	Scheme *D*
C432-223	0.121 3	0.161 6	0.085 7	0.087 1
C499-753	0.111 6	0.184 6	0.095 2	0.094 3
C880-767	0.035 1	0.063 6	0.016 3	0.048 7
C2670-2387	0.102 4	0.177 5	0.095 5	0.168 1
C3540-3195	0.130 2	0.146 3	0.128 5	0.136 7
C5315-4737	0.148 8	0.188 8	0.017 6	0.067 6
C6288-1901	0.017 7	0.032 9	0.017 3	0.027 8
C7552-10112	0.048 0	0.067 4	0.040 1	0.059 8

5 Conclusions

Identifying reliability-critical gates in a circuit structure is an important step for improving circuit reliability at a low cost and is also an important aim to evaluate the reliability of the circuit structure. Our method PGA gives an embedded parallel-structured operation for crossover and mutation of the GA to avoid unnecessary operations. To significantly reduce the search space, an initial population consisting of dominant individuals is constructed. The operations of PRC and PRM are to reduce the probability of the dominant individuals being destroyed and improve the local search capability of the GA, and the PRS operation to preserve dominant individuals and escape from local traps are defined. It adopts a diversity analysis for the population to avoid unnecessary calculations and alleviate premature stagnation. These are the key techniques to promote fast convergence and prevent premature convergence. Furthermore, we proposed a CGR calculation method for gates to provide stable searching performance. The simulation results on benchmark circuits demonstrated the advantages of the proposed method in terms of accuracy and efficiency. Hence, this method may play an important role in improving circuit reliability at a low cost.

References

[1] Xiao J, Jiang J H, Li X X *et al.* A novel trust evaluation method for logic circuits in IoT applications based on the E-PTM model. *IEEE Access*, 2018, 6: 35683-35696.

[2] Zhang Y, Li H W, Li X W. Automatic test program generation using executing-trace-based constraint extraction for embedded processors. *IEEE Transactions on Very Large Scale Integration Systems*, 2013, 21(7): 1220-1233.

[3] Srinivasu B, Sridharan K. A transistor-level probabilistic approach for reliability analysis of arithmetic circuits with applications to emerging technologies. *IEEE Transactions on Reliability*, 2017, 66(2): 440-457.

[4] Xiao J, Lee W, Jiang J H *et al.* Sensitivity evaluation of input vectors with masking effects in digital circuits. *Chinese Journal of Computers*, 2018, 41(10): 2282-2294. (in Chinese)

[5] Bickford J, Habib N, Li B *et al.* Integrated circuit chip reliability qualification using a sample-specific expected fail rate. United States Patent, http://www.freepatentsonline.com/9639645.pdf, May 2019.

[6] Han J, Hao C, Liang J H *et al.* A stochastic computational approach for accurate and efficient reliability evaluation. *IEEE Transactions on Computers*, 2014, 63(6): 1336-1350.

[7] Mittal S, Vetter J S. A survey of techniques for modeling and improving reliability of computing systems. *IEEE Transactions on Parallel and Distributing Systems*, 2016, 27(4): 1226-1238.

[8] Xiao J, Lee W, Jiang J H *et al.* Circuit reliability estimation based on an iterative PTM model with hybrid coding. *Microelectronics Journal*, 2016, 52: 117-123.

[9] Bernardini A, Schlichtmann U. Symbolic fault modeling and model counting for the identification of critical gates in digital circuits. In *Proc. IEEE GMM/ITG/ GI-Symposium Reliability by Design*, Sept. 2015, pp.15-22.

[10] Xiao J, Lou J G, Jiang J H *et al.* Blockchain architecture reliability-based measurement for circuit unit importance. *IEEE Access*, 2018, 6: 15326-15334.

[11] Ibrahim W. Identifying the worst reliability input vectors and the associated critical logic gates. *IEEE Transactions on Computers*, 2016, 65(6): 1748-1760.

[12] Ibrahim W, Beiu V, Amer H. Why should we care about input vectors? In *Proc. the 2009 Joint IEEE North-East Workshop on Circuits and Systems and TAISA Conference*, Jun. 2009, Article No. 60.

[13] Janssen H. Monte-Carlo based uncertainty analysis: Sampling efficiency and sampling convergence. *Reliability Engineering & System Safety*, 2013, 109: 123-132.

[14] Rubinstein R Y, Kroese D P. Simulation and the Monte Carlo Method (3rd edition). Wiley, 2016.

[15] Han J, Chen H, Boykin E *et al.* Reliability evaluation of logic circuits using probabilistic gate models. *Microelectronics Reliability*, 2011, 51(2): 468-476.

[16] Tang A, Jha N K. GenFin: Genetic algorithm-based multiobjective statistical logic circuit optimization using incremental statistical analysis. *IEEE Transactions on Very Large Scale Integration Systems*, 2016, 24(3): 1126-1139.

[17] Srinivas M, Patnaik L. On generating optimal signal probabilities for random tests: A genetic approach. *VLSI Design*, 1996, 4(3): 207-215.

[18] Nagamani A, Nayak A, Nanditha N *et al.* A genetic algorithm based heuristic method for test set generation in reversible circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018, 37(2): 324-336.

[19] Liliakadri R, Boctor F. An efficient genetic algorithm to solve the resource-constrained project scheduling problem with transfer times: The single mode case. *European Journal of Operational Research*, 2018, 265(2): 454-462.

[20] Kvassay M, Zaitseva E, Levashenko V *et al.* Reliability analysis of multiple-outputs logic circuits based on structure function approach. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017, 36(3): 398-411.

[21] Xiao J, Lou J G, Jiang J H. A fast and effective sensitivity calculation method for circuit input vectors. *IEEE Transactions on Reliability.* doi:10.1109/TR. 2019.2897455.

[22] Corus D, Dang D, Eremeev A *et al.* Level-based analysis of genetic algorithms and other search processes. *IEEE Transactions on Evolutionary Computation*, 2018, 22(5): 707-719.

[23] Bito J, Jeong S, Tentzeris M. A novel heuristic passive and active matching circuit design method for wireless power transfer to moving objects. *IEEE Transactions on Microwave Theory Techniques*, 2017, 65(4): 1094-1102.

[24] Ye Z, Li Z, Xie M. Some improvements on adaptive genetic algorithms for reliability-related applications. *Reliability Engineering & System Safety*, 2017, 95(2): 120-126.

[25] Silver D, Huang A, Maddison C J *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016, 529(7587): 484-489.

[26] Sivaraj R, Ravichandran D. A review of selection methods in genetic algorithms. *International Journal of Engineering Science and Technology*, 2011, 3(5): 3792-3797.

[27] Pandey H M, Shukla A, Chaudhary A *et al.* Evaluation of genetic algorithm's selection methods. In *Proc. the 3rd International Conference on Information Systems Design and Intelligent Applications*, January 2016, pp.731-738.

[28] Krishnaswamy S, Viamontes G F, Markov I L *et al.* Accurate reliability evaluation and enhancement via probabilistic transfer matrices. In *Proc. the 2005 Design, Automation and Test in Europe*, March 2005, pp.282-287.

[29] Ji Z, Xia Q, Meng G. A review of parameter learning methods in Bayesian network. In *Proc. the 11th International Conference on Intelligent Computing*, *Part III*, Aug. 2015, pp.3-12.

[30] Shahani A, Natu N, Badami A. A Genetic Algorithm for VLSI Floorplanning: An Intuitive Approach to Optimisation and Miniaturisation of IC. LAP LAMBERT Academic Publishing, 2012.

Jie Xiao received his Ph.D. degree in computer system architecture from Tongji University, Shanghai, in 2013. He is currently working with the College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou. His current research interests include reliability evaluation and fault-tolerant design, deep learning and combinatorial optimization-computation. He also serves as a consultant and technical adviser for a research institute in electronic information fields. He is a member of CCF.

Zhan-Hui Shi does his postgraduate work at the College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou. His current research interests include reliability evaluation and fault-tolerant design, deep learning and combinatorial optimization computation.

Jian-Hui Jiang received his B.E., M.E. and Ph.D. degrees in traffic information engineering and control from Shanghai Tiedao University (in April 2000, it was merged to Tongji University), Shanghai, in 1985, 1988, and 1999, respectively. From 1994-2000, he was an associate professor of computer science and technology at Shanghai Tiedao University, Shanghai. Since 2000, he has been a full professor of computer science and technology at Tongji University, Shanghai. From 2007-2011, he was chair of the Department of Computer Science and Technology at Tongji University, Shanghai. Since 2011, he is associate dean of the School of Software Engineering at Tongji University, Shanghai. He has served on several program committees of national or international symposiums or workshops including IEEE Pacific Rim International Symposium on Dependable Computing, IEEE Asian Test Symposium, IEEE Workshop on RTL and High-Level Testing. He has coauthored two books and published more than 200 technical papers. His current research interests include dependable systems and networks, software reliability engineering, VLSI/SoC testing and fault-tolerance.

Xu-Hua Yang received his B.E. degree in automation from China University of Petroleum, Dongying, in 1993, and his M.S. degree and Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, in 2001 and 2004, respectively. He is currently a professor with the College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou. His current research interests include artificial intelligent, complex network system, intelligent transportation system, link prediction, and deep learning.

Yu-Jiao Huang received her B.S. degree in information and computer science, M.S. degree in computational mathematics, and Ph.D. degree in control theory and control engineering from Northeastern University, Shenyang, in 2008, 2010 and 2014, respectively. She is now a lecturer at Zhejiang University of Technology, Hangzhou. Her research interests are in areas of artificial neural networks, stability theory, and dynamical systems.

Hai-Gen Hu received his Ph.D. degree in control theory and control engineering from Tongji University, Shanghai, in 2013. He is currently an associate professor in the College of Computer Science and Technology at Zhejiang University of Technology, Hangzhou. And he is currently also a postdoctoral researcher in LITIS Laboratory (laboratoire d'informatique, de traitement de l'information et des systèmes), Université de Rouen, France. His current research interests include machine learning (deep learning), neural networks, multi-objective evolutionary algorithms, greenhouse environmental control, bioinformatics, and their applications.