Available online at www.sciencedirect.com

SCIENCE DIRECT®

Advanced Engineering Informatics 20 (2006) 7–21

ADVANCED ENGINEERING INFORMATICS

www.elsevier.com/locate/aei

A hybrid real-parameter genetic algorithm for function optimization

Shun-Fa Hwang [a,*], Rong-Song He [a,b]

[a] *Institute of Engineering Technology, National Yunlin University of Science and Technology, 123 University Road, Sec. 3, Touliu, Taiwan 640, ROC*
[b] *Department of Mechanical Engineering, Wu Feng Institute of Technology, 117 Chian-Kuo Road, Sec. 2, Ming-Hsiung, Chia-Yi, Taiwan 621, ROC*

Received 18 February 2005; revised 31 August 2005

Abstract

One drawback of genetic algorithm is that it may spend much computation time in the encoding and decoding processes. Also, since genetic algorithm lacks hill-climbing capacity, it may easily fall in a trap and find a local minimum not the true solution. In this paper, a novel adaptive real-parameter simulated annealing genetic algorithm (ARSAGA) that maintains the merits of genetic algorithm and simulated annealing is proposed. Adaptive mechanisms are also included to insure the solution quality and to improve the convergence speed. The performance of the proposed operators has been discussed in detail and compared to other operators, and the performance of the proposed algorithm is demonstrated in 16 benchmark functions and two engineering optimization problems. Due to their versatile characteristics, these examples are suitable to test the ability of the proposed algorithm. The results indicate that the global searching ability and the convergence speed of this novel hybrid algorithm are significantly better, even though small population size is used. Also, the proposed algorithm has good application to engineering optimization problems. Hence, the proposed algorithm is efficient and improves the drawbacks of genetic algorithm.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Genetic algorithm; Simulated annealing; Adaptive mechanism; Function optimization; Design optimization

1. Introduction

Genetic algorithm (GA) is an important stochastic search algorithm for solving optimization problems during the last decade. It has been widely and successfully applied to various problems like control problem [1–3], image processing [4], and path planning in construction sites [5]. GA is different from most conventional calculus-based search algorithms in the following characteristics: no limitation on the continuity or discreteness of the search space, parallel computation of a population of solutions, using natural selection criteria, and no gradient information.

However, one major drawback of GA is its premature convergence to local minima resulting in low accuracy. To avoid the premature convergence, one way is to use large population, but its computation time is a burden and the convergence speed to obtain results with reasonable precision is slow. Under the restriction of small population, four possible ways were presented. The first one is to revise the gene operators. Leite and Topping [6] proposed an effective one-point crossover model, and Hasançebi and Erbatur [7] proposed two crossover models, which were called mixed crossover technique and direct design variable exchange crossover technique. Their results were better than those by traditional models, but these improved crossover models were proposed for simple GA (SGA) and not suitable for real-parameter GA (RGA). Griffiths and Miles [8] applied advanced two-dimensional gene operators to search the optimal cross-section of a beam and significantly improved the results. The second way is to adjust gene probability. Leite and Topping [6] adopted a variable mutation probability and obtained an outperformed result. Hsieh et al. [9] used Taguchi method to find the optimal operating parameters in GA to improve its performance. The third way modifies the fitness scaling technique. Nanakorn and Meesomklin [10] used a bilinear fitness scaling technique to encourage the solution to move toward the feasible region. Wong and Hamouda [11] proposed a new fitness scaling technique that is called fitness mapping and provided better results than a conventional one. The fourth way is to merge GA with other techniques. Jeong and Lee [2] proposed an adaptive simulated annealing (SA) mutation to replace the traditional mutation of GA and the other operators were the same as SGA. It was applied to system identification, and the convergence speed to the desired solution was improved. Adler [12] used SA instead of the normal crossover operator and normal mutation operator, and the population size

* Corresponding author. Tel.: +886 5 5342601x4143; fax: +886 5 5312062.
E-mail address: hwangsf@yuntech.edu.tw (S.-F. Hwang).

1474-0346/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.aei.2005.09.001

could be small. Since his method is like SA more than GA, it has the same drawbacks as SA, like slow convergence speed. Tan et al. [13] applied a complete SA process to replace the mutation process of GA for system identification. Although it improved the solution quality, the computation time was significantly increased. Brown et al. [14] presented a parallel genetic heuristics consisting of GA stage and SA stage for a quadratic assignment problem. Since all the above combined methods were based on SGA, they still have the drawbacks of SGA. Shapiro [15] used fuzzy logic to determine the values of gene parameters in each generation, and the performance of GA was improved. Leung and Wang [16] used orthogonal genetic algorithm with quantization (OGA/Q) to obtain global minima. They added the process between GA operators such that their method is complicated.

Soltani et al. [5] concluded that GA always finds the optimal or near-optimal solution and the main drawback of GA is time-consuming in the fine-tuning process. Two possible reasons are as follows. (1). The crossover probability is not chosen appropriately because the crossover process is a very important operator of GA. (2) When the mutation probability is much less than the crossover probability, the searched solution is difficult to move to other search spaces. To overcome these problems, Adler [12] used SA to replace the gene operators of GA, but it is a strong resemblance to SA. Jeong and Lee [2] used SA to replace the mutation operator and SGA still played a major role. Hence, it has the same drawbacks as SGA. Shapiro [15] used fuzzy to decide crossover probability and mutation probability in each generation. However, it is very complicated.

To improve the gene operators of GA, a novel hybrid algorithm that is called adaptive real-parameter simulated annealing genetic algorithm (ARSAGA) is proposed in this work. This novel hybrid algorithm is based on a real-parameter genetic algorithm (RGA) with a hybrid crossover operator, a hybrid mutation operator composing of SA mutation and GA mutation, and adaptive mechanisms to determine the dynamic gene probability. It is designed to improve GA's efficacy such as the hill-climbing ability, the convergence speed to the global optimum solution or near global optimum solution, and the reliability as well as accuracy of the searched solution. Some well-known benchmark functions [16–20] that may be discontinuous or non-convex, multi-modal, high dimensional, or even not differentiable are used to test this novel hybrid algorithm. Furthermore, two engineering optimization problems including a welded beam and a 15-bar truss are utilized to test the applicability of the proposed algorithm.

2. Adaptive real-parameter simulated annealing genetic algorithm

2.1. Real-parameter genetic algorithm

One major drawback of a simple genetic algorithm is that it encodes parameters as finite-length strings such that much computation time is wasted in the encoding and decoding processes. Hence, a real-parameter genetic algorithm (RGA) is proposed to overcome this problem. Instead of the coding processes, RGA directly operates on the parameters and much computation time is saved. As for the genetic operators, RGA is the same as SGA in the reproduction process, but they may be different in the crossover and mutation processes.

The reproduction process utilizes the Darwinian principle of 'survival of the fittest'. By defining a normalized fitness, reproduction process evaluates each chromosome's fitness and uses the well-known roulette selection criterion to produce the next population. The chromosome (solution) with a higher normalized fitness has a higher survival probability into the next population.

In the crossover process of RGA, two solutions called parents are randomly selected from a mating pool. Crossover process is applied if a random value generated between 0 and 1 is not larger than the crossover probability P_c. Then, the genetic operation of sexual recombination is applied to the pairs of genes (parameters) of the parents. Because the parameter pair for the crossover process is randomly selected, it is important to ensure that the new parameters are within the search domain. In general, there are three genetic operation modes of sexual recombination: (1) one-point crossover, (2) two-point crossover, and (3) uniform crossover.

Suppose that the two parents could be represented as $y_1=\{a_1,a_2,\ldots,a_n\}$ and $y_2=\{b_1,b_2,\ldots,b_n\}$, where the parameters a and b are real numbers and n is the total number of the parameters to be searched. The one-point crossover denoted as mode 1 in this work has only one pair of parameters for the crossover process. In this mode, a random number r_{random} whose value is between 0 and 1 is generated, and if its value does not exceed the crossover probability P_c, then one crossover point is randomly generated. If the crossover point is i, for example, then the new solutions are

$$y_1' = \{a_1, a_2, \ldots, a_i', b_{i+1}, \ldots, b_n\} \tag{1}$$

$$y_2' = \{b_1, b_2, \ldots, b_i', a_{i+1}, \ldots, a_n\} \tag{2}$$

where

$$a_i' = \alpha_i a_i + (1-\alpha_i) b_i \tag{3}$$

$$b_i' = \alpha_i b_i + (1-\alpha_i) a_i \tag{4}$$

where α_i is a random number and $0\le\alpha_i\le 1$. It is noted that Eqs. (3) and (4) are used for real-parameters. The two-point crossover is referred to as mode 2. Similar to mode 1, if the random number r_{random} does not exceed the crossover probability P_c, two crossover points are randomly generated. If the crossover points are i and j, all the parameter pairs between i and j have to go through the crossover process and the new solutions are

$$y_1' = \{a_1, a_2, \ldots, a_i', a_{i+1}', \ldots, a_j', \ldots, a_n\} \tag{5}$$

$$y'_2 = \{b_1, b_2, \ldots, b'_i, b'_{i+1}, \ldots, b'_j, \ldots, b_n\} \tag{6}$$

The expressions of the pairs (a'_i, b'_i), $(a'_{i+1}, b'_{i+1})\ldots$ and (a'_j, b'_j) are the same as Eqs. (3) and (4). The uniform crossover is called mode 3 in this work. The crossover process is applied to every pair (a_i, b_i) of the parents if the random number r_{random} does not exceed the crossover probability. The expressions of the new pairs (a'_i, b'_i) are the same as Eqs. (3) and (4).

In addition, a novel crossover mode is proposed as mode 4 here for trying to improve the performance of RGA. In this mode, if the random number r_{random} does not exceed the crossover probability, then a random crossover point is generated and the crossover process is done from this selected point to either the end point n or the first point 1 of the solution, which is randomly selected. For example, if the generated number is i and the end point n is chosen, the new solutions are

$$y'_1 = \{b_1, b_2, \ldots, a'_i, a'_{i+1}, \ldots, a'_j, \ldots, a'_n\} \tag{7}$$

$$y'_2 = \{a_1, a_2, \ldots, b'_i, b'_{i+1}, \ldots, b'_j, \ldots, b'_n\} \tag{8}$$

The expressions for the pairs of (a'_j, b'_j), where $j = i–n$, are the same as Eqs. (3) and (4). This mode could be considered as a hybrid mode of one-point and two-point crossover modes, and it is more like mode 2 than mode 1. Because this hybrid mode forces suitable amount of parameter pairs to go through the crossover process, it increases the diversity of solution such that a better solution could be found from other spaces in a reasonable time.

As for the mutation operation of RGA, the uniform mutation mode is adopted. Each parameter of a solution is to be mutated or not according to its random number r_{random} whose value is between 0 and 1. If r_{random} does not exceed the mutation probability P_{m}, then the mutation process is executed. For example, one solution is represented as $y'_1 = \{a'_1, a'_2, \ldots, a'_n\}$ that is taken from the population after the crossover process. When a parameter a'_i of the parent y'_1 is selected to be mutated, its new parameter value after GA mutation is

$$a''_i = \alpha_i a'_i + (1 - \alpha_i) b'_i \tag{9}$$

where α_i is a random number, $0 \le \alpha_i \le 1$, and b'_i is the ith parameter of $y'_2 = \{b'_1, b'_2, \ldots, b'_n\}$ that is randomly chosen from the population after the simulated annealing process.

2.2. Simulated annealing

SA is the simulation of annealing of a physical many-particle system for finding the global optimum solutions of a large combinatorial optimization problem. Its original theory is briefly described as follows. Suppose that a system is allowed to reach thermal equilibrium at temperature T. It could be characterized by the Boltzmann distribution $\text{Pro}_{(q)}$ with the energy E_q in a state q as

$$\text{Pro}_{(q)} = \frac{1}{Z_{(T)}} \mathrm{e}^{(-E_q/K_\mathrm{B}T)} \tag{10}$$

where K_B is the Boltzmann constant. The partition factor $Z_{(T)}$ is defined as

$$Z_{(T)} = \sum_q \mathrm{e}^{(-E_q/K_\mathrm{B}T)} \tag{11}$$

where the summation runs over all possible macroscopic states q.

If a system is in a configuration q with an internal state variable vector $\tilde{s}_q$ at time t, then a new configuration r of the system with an internal state variable vector $\tilde{s}_r$ at time $t+1$ is generated randomly. The Boltzmann distribution of the configuration r is represented as

$$\text{Pro}_{(r)} = \mathrm{e}^{-(E_r - E_q/K_\mathrm{B}T)} \tag{12}$$

If $E_r - E_q < 0$, then configuration r is accepted as the next configuration at time $t+1$. If $E_r - E_q \ge 0$, the probability of acceptance of this new configuration is given by $\text{Pro}_{(r)}$. That is to say, the system state may be on a higher energy state. This acceptance rule for next configurations is referred to as the Metropolis criterion.

For a combinatorial optimization problem, the configuration q at time t is accepted and the probability distribution is given by

$$\text{Pro}_{(q)} = \frac{1}{Q_{(q)}} \mathrm{e}^{(-c_{(q)}/C_\mathrm{P})} \tag{13}$$

where $c_{(q)}$ is the energy function, C_p is the control parameter, and $Q_{(q)}$ is a normalized constant depending on the control parameter C_P. A new configuration r at time $t+1$ can be randomly chosen from the neighborhood of q. The probability of configuration r to be the next configuration is decided by the Metropolis criterion. This process is continued until equilibrium is reached as the time is increased from current value to infinity. Then the probability distribution of the configurations of a combinatorial optimization problem approaches the Boltzmann distribution. Thus, the energy function $c_{(q)}$ and the control parameter C_p take the roles of the energy and temperature, respectively.

Besides, one needs an annealing process to obtain a lower-energy configuration. In the annealing schedule, a sufficiently high temperature needs to be provided, and a cooling process is to lower slowly the temperature such that there is enough time to reach the corresponding equilibrium state at each temperature. The well-known cooling schedule that provides the necessary and sufficient conditions for convergence is

$$C(t) = \frac{C_0}{\log t} \quad \forall\, t > 0 \tag{14}$$

where $C(t)$ is a given sequence of control parameter (temperature), C_0 is a constant, and t denotes the time. When t approaches infinity, $C(t)$ approaches zero. When the annealing schedule is applied to an optimization problem, the energy/cost function becomes the objective function, and the configuration of the internal state variables becomes the configuration of the searched parameters of the optimization problem.

2.3. Adaptive real-parameter simulated annealing genetic algorithm

A novel hybrid algorithm that merges RGA with SA is proposed in this section and called adaptive real-parameter simulated annealing genetic algorithm (ARSAGA). This novel hybrid algorithm maintains the merit of RGA by using its crossover process and merges the merit of SA with the original mutation process of RGA. Also, adaptive mechanisms are included to improve the searching ability for optimum solutions.

In the annealing schedule of ARSAGA, Eq. (14) is transformed to the form of Eq. (15) and Eq. (13) is converted to Eq. (16).

$$T_{(l)} = \frac{T_0}{\log l} \tag{15}$$

$$\mathrm{Pro}_{(l)} = \mathrm{e}^{-(E_\mathrm{r}-E_\mathrm{q}/T_{(1)})} \tag{16}$$

where l denotes as an integer time step, T_0 is an initial constant temperature, $T_{(\mathrm{l})}$ is a temperature sequence. The objective function values of the original solution and the new solution are represented as E_q and E_r, respectively. It is noted that $\mathrm{Pro}_{(l)}$ equals 1 when $l=1$. From Eqs. (15) and (16), one can obtain that

$$\mathrm{Pro}_{(l)} = l^{(-(E_\mathrm{r}-E_\mathrm{q})/T_0)} \tag{17}$$

The SA mutation operator operates as follows. Generate randomly a new solution from the neighborhood of the original solution that is obtained from the population after the crossover process. If the new value of the objective function is less than the original one, that is $E_\mathrm{r}-E_\mathrm{q}<0$, the new solution is better than the original one and it is accepted. On the other hand, if $E_\mathrm{r}-E_\mathrm{q}\geq 0$, the new one is accepted only when its acceptance probability $\mathrm{Pro}_{(l)}$ given in Eq. (17) is greater than a random value generated between 0 and 1. There has been much work done about the choosing of the constant T_0 in Eq. (17). However, it is still difficult to determine T_0 because it is dependent on the strategies used for different problems. In general, T_0 can be represented as a function of $f_{\max}$ and $f_{\min}$, which represents the maximum and minimum objective function values of the initial population, respectively. In this work, for an objective function to be minimized, T_0 could be simplified to

$$T_0 = |f_{\min}| \tag{18}$$

Similarly, for an objective function to be maximized, T_0 could be chosen as

$$T_0 = |f_{\max}| \tag{19}$$

When the best solution keeps the same for some consecutive generations, the executed algorithm may be frozen at a local minimum, and some changes should be done on the searching strategy of the algorithm to prevent premature convergence. Adaptive mechanisms are proposed here to do the change and for the following reasons. First, when the best solution does not change, the evolutionary speed of population is very slow or even zero. Hence, the crossover probability should be increased to generate some new different solutions from other solution spaces to avoid premature convergence. Secondly, the normal mutation probability is too small to escape a trap. Adjusting the mutation probability could find the solution from other search spaces to improve the hill-climbing capability and more likely to find a better solution. Based on the above reasons, the adaptive mechanism must satisfy the following conditions. (1) When the repeated generation number of the best solution increases to a given value, the probability of gene operators must increase. (2) The probability of gene operators should increase as the increasing of the repeated generation number of the best solution. (3) When the best solution is changed, the probabilities of the gene operators are reset to their initial values. (4) The probability must be constrained in a suitable range, which has been suggested by some researchers [19–21]. Therefore, simple adaptive mechanisms are proposed here. If the best solution is the same for the last N consecutive generations and $N>N_\mathrm{frozen}$, the crossover probability and mutation probability are changed according to the following two equations.

$$P_\mathrm{c} = P_\mathrm{c0} + \frac{N-N_\mathrm{frozen}}{N}(\alpha - P_\mathrm{c0}) \tag{20}$$

$$P_\mathrm{m} = P_\mathrm{m0} + \frac{N-N_\mathrm{frozen}}{N}(\beta - P_\mathrm{m0}) \tag{21}$$

where the frozen number N_frozen is a positive integer. The initial crossover probability and the initial mutation probability are represented as P_c0 and P_m0, respectively. Besides, α and β are constant real numbers. If $N\leq N_\mathrm{frozen}$ or the fittest solution changes such that N is reset to zero, the crossover probability and the mutation probability are reset to their initial values as

$$P_\mathrm{c} = P_\mathrm{c0} \tag{22}$$

$$P_\mathrm{m} = P_\mathrm{m0} \tag{23}$$

Eqs. (20) and (22) are called adaptive crossover mechanism of RGA, and Eqs. (21) and (23) are called adaptive mutation mechanism of RGA. If there is no adaptive mechanism included in the algorithm, for example in BGA or RGA, the crossover probability and the mutation probability will remain the same as their initial values as shown in Eqs. (22) and (23). To improve the fine-tuning ability of RGA, SA has been used in the mutation process of RGA. When $N>N_\mathrm{frozen}$, the l value in Eq. (17) must be reset to 1. In addition, the elitist strategy, by which the best solution of each generation is copied to the next generation, is adopted here to insure the solution quality.

The flow chart of ARSAGA is shown in Fig. 1 and is described as follows:

Step 1 Set the values of the parameters as listed in Table 1.
Step 2 Generate a population with a large size M_I and calculate the objective function value and find the fitness of each solution.

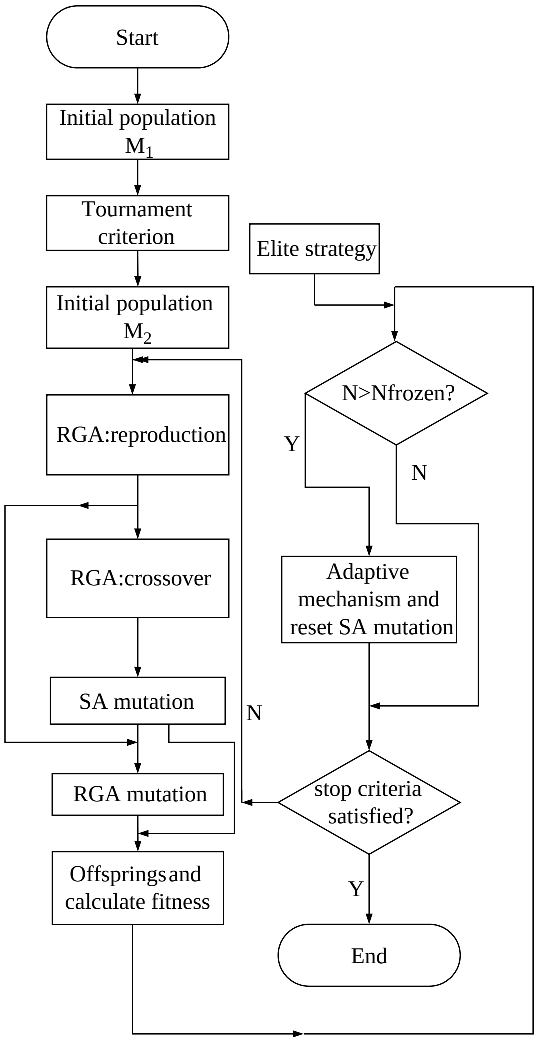

Fig. 1. Flow chart of ARSAGA (a) F_1; (b) F_2; (c) F_3; (d) F_4; (e) F_5; (f) F_6; (g) F_7.

Step 3 Use the tournament criterion to select the population with a small size M_2 from the large population.
Step 4 Apply reproduction process by the roulette wheel criterion.
Step 5: Apply crossover process. Note that only a half population goes to step 6 in order to reduce the number of function evaluations.
Step 6 Apply SA process and this half population simultaneously goes to step 7 and step 8.
Step 7 A half population obtained from Step 4 and a half population obtained from step 6 are mutated by the mutation probability of RGA to produce just a half population as described in Eq. (9).
Step 8 Calculate the objective function value and find the fitness of the offspring.
Step 9 If the best solution is not improved, the elitist strategy is adopted.
Step 10 If the best solution is the same for N consecutive generations and $N > N_{\text{frozen}}$, then adaptive mechanisms start and SA process is reset.
Step 11 If the stop criterion is satisfied, then algorithm ends. Otherwise, go to step 4.

Table 1
Parameter values used in ARSAGA

Parameter	Value
Generation number	500
Population size	40
Initial crossover probability P_{c0}	0.7
Initial mutation probability P_{m0}	0.01
Constant α	0.9
Constant β	0.2

3. Verification examples

3.1. Parameter values of ARSAGA

In the proposed algorithm, the values of the crossover probability P_c and the mutation probability P_m need to be selected appropriately. Although each parameter has been suggested in several papers [21–23], some trials are necessary to select them simultaneously and for the specific problems. The parameter values determined after some tests are listed in Table 1, and they will be adopted in the following sections.

3.2. Performance of the proposed operators

In this section, seven functions are utilized to test the capability of the proposed operators in GA-based methods. These functions selected from several articles [16,21,24–27] are listed in Table 2. It's noted that the first five functions are to be minimized and the remainders are to be maximized.

3.2.1. Comparison of different crossover modes

In addition to the three common modes of crossover process, the proposed crossover mode is applied to the above functions to compare their efficiency on function optimization. Fig. 2 shows the average values and the best values of the objective function among 30 independent simulations by RGA with these four crossover modes. The average value is calculated from the 30 independent simulations, and the best value means the smallest value among them. Note that the mode number in the abscissa of Fig. 2 represents the corresponding crossover mode as listed in Table 3. From the results of the best values, they indicate that even though the problem is complex or the search space is large, the proposed crossover mode that is denoted as mode 4 always has better solutions than the other three crossover modes. In other words, the proposed crossover mode could find the true solution or near the desired solution for different functions with different characteristics. Considering the results of the average values, the average values found by mode 4 are clearly better than those found by the other three modes. This may imply that the solutions of the proposed crossover mode are more reliable than those of the other three modes. Therefore, the proposed crossover mode has better performance than the other three traditional modes. This may be attributed to that the proposed

Table 2
Seven simple test functions

Function number	Function	Feasible solution space
F_1	$f(x)=\sum_{i=1}^{4}(x_i^2-8)^2+5\sum_{i=1}^{4}x_i+57.3276$	$-3.0\leq x_i\leq 3.0$
F_2	$f(x_i)=100(x_1^2-x_2)^2+(1-x_i)^2$	$-2.048\leq x_i\leq 2.048$
F_3	$f(x_i)=\sum_{i=1}^{5}\text{int}(x_i)$	$-5.12\leq x_i\leq 5.12$
F_4	$f(x_i)=\sum_{i=1}^{50}i\,x_1^4+\text{gauss}(x_i,0,1)$	$-1.28\leq x_i\leq 1.28$
F_5	$f(x_i)=\sum_{i=1}^{10}-x_i\sin(\sqrt{\lvert x_i\rvert})$	$-500\leq x_i\leq 500$
F_6	$f(x_i)=100(x_1^2-x_2)^2+(1-x_1)^2c$	$-2.048\leq x_i\leq 2.048$
F_7	$f(u(k))=6\sin(\pi u(k))+3\sin(3\pi u(k))+\sin(5\pi u(k))$ $u(k)=\sin(2\pi k/250)$	$275\leq k\leq 375$ k is an integer number

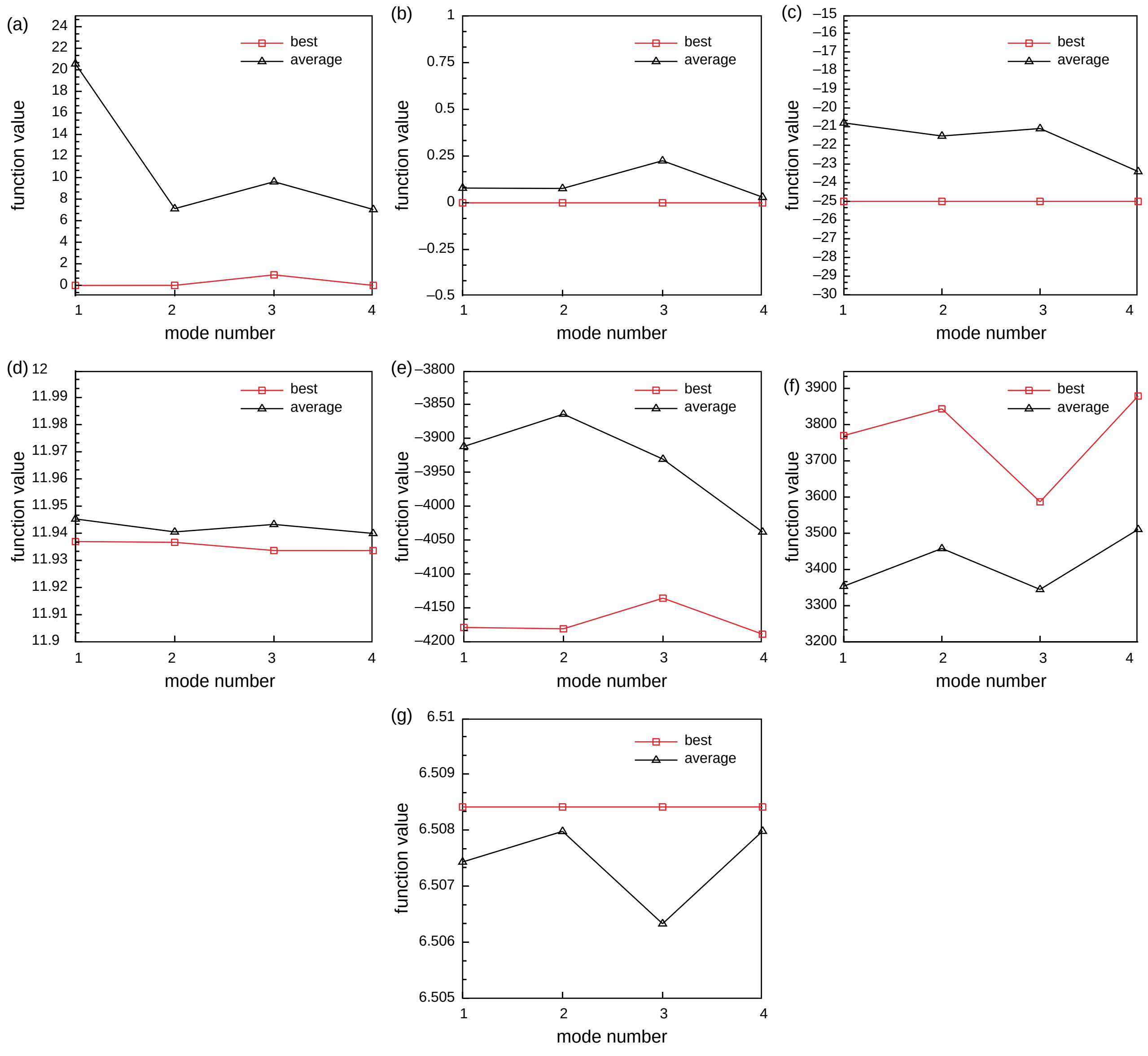

Fig. 2. Values of seven functions by different crossover modes in RGA. (a) F_1; (b) F_2; (c) F_3; (d) F_4; (e) F_5; (f) F_6; (g) F_7.

Table 3
The four crossover modes

Mode number	Crossover mode
1	One-point crossover mode
2	Two-point crossover mode
3	Uniform crossover mode
4	The proposed crossover mode

crossover mode reasonably increases the diversity of the solutions such that they can escape from traps to other search spaces. For this reason, it could obtain a better solution than the other three crossover modes.

3.2.2. Performance of adaptive mechanisms and SA

Algorithm indexes 1–8 in the abscissa of Fig. 3 represent that the function value is obtained by ARSAGA with $N_{\text{frozen}} =$ 1, 2, 5, 10, 20, 30, 50, and 100, respectively. The smaller the frozen number is, the easier the adaptive mechanism is triggered. Algorithm index 9 denotes that the results are found by the combination of RGA with SA. That is to say, adaptive mechanisms are not used in the method of algorithm index 9 that is also called RSAGA in this work. For comparison, algorithm index 10 is just the plain RGA without including both SA and adaptive mechanisms. Comparing the results of algorithm indexes 6–10 indicates that the plain RGA always has the worst function values, and algorithm index 9 always has better results than RGA. This should be attributed to the assistance of SA. Under the assistance of adaptive mechanisms, algorithm index 8 could find better solutions than algorithm index 9, even though its frozen number N_{frozen} is equal to 100. Among algorithm indexes 6–8, the function values are improved as the frozen number is decreased. From algorithm indexes 3–6, the found function values are almost the

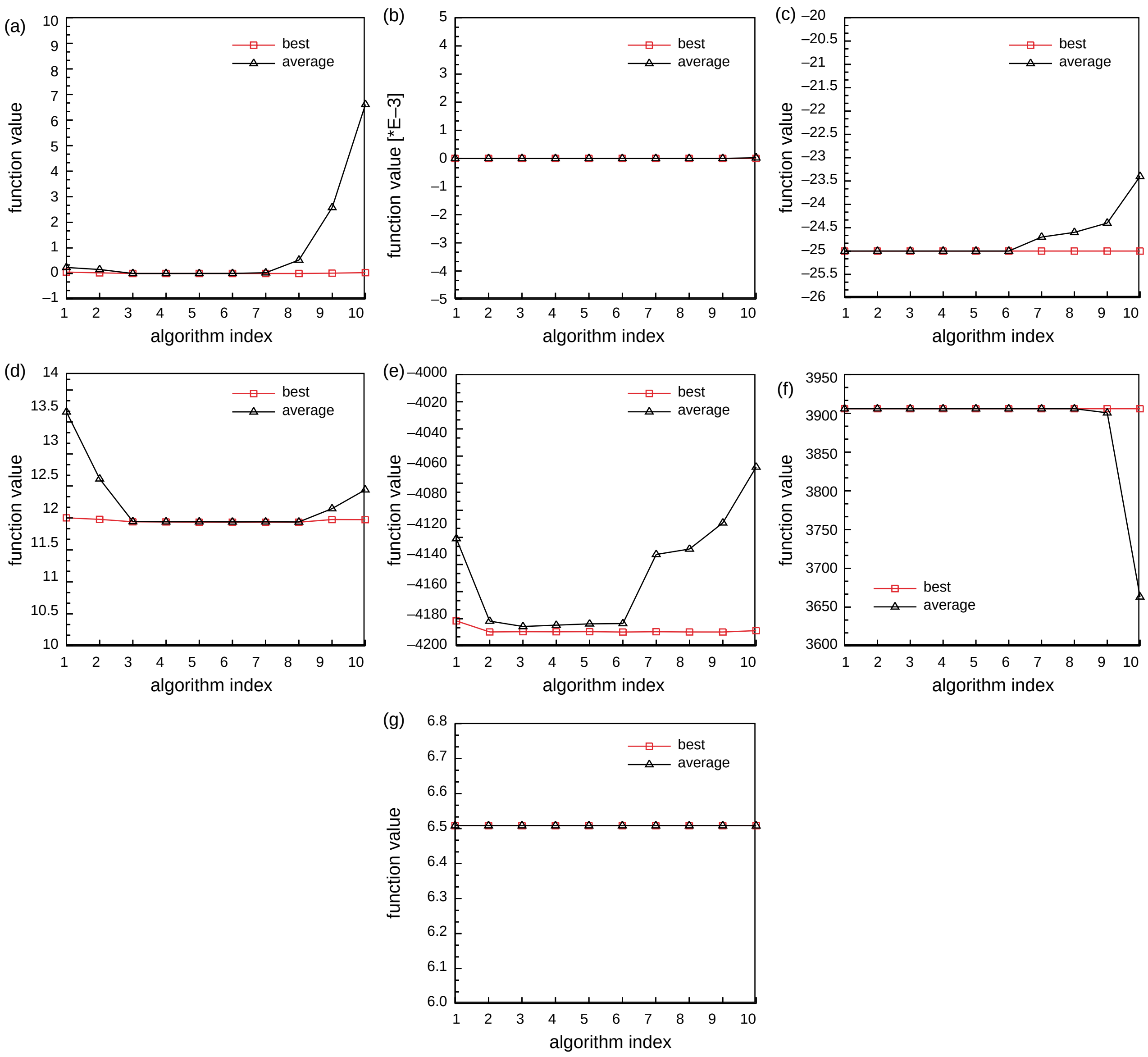

Fig. 3. Values of seven functions by different GAs. (a) f_1; (b) f_2; (c) f_3; (d) f_4; (e) f_5; (f) f_6; (g) f_7; (h) f_8; (i) f_9; (j) f_{10}; (k) f_{11}; (m) f_{12}; (n) f_{13}; (o) f_{14}; (p) f_{15}; (q) f_{16}.

same, and it may imply that the stable frozen numbers are from 5 to 30. If one considers algorithm index 1 or 2 in which the frozen number is 1 or 2, the results become worse. For example, consider F_5 whose acceptable value is -4180.0. From Fig. 3(e), the best objective function value among 30 independent simulations by most algorithms is less than -4189.0, but the average objective function value is less than the acceptable value only when the frozen number is from 5 to 30. That is to say, one could obtain a reliable solution, only if the frozen number is properly chosen. As for some simple functions like F_6 and F_7, whether the frozen number is large or small, the found solutions are in reasonable precision.

From the above results, it shows that the plain RGA is not powerful because the found solution is far from the true value. The combination of RGA with SA could find better function values. Moreover, the further addition of adaptive mechanisms could continue to improve the function values. Because of SA and adaptive mechanisms, the algorithm has better viability than other RGAs and improves the drawbacks of RGA. To have good and stable performance, the frozen number should be in a suitable range. When the frozen number is too large, adaptive mechanisms have little effects on the solution quality and the proposed algorithm may not escape from a trap. In this situation, ARSAGA is just like RSAGA. When a very small frozen number is chosen, the results may be not good enough for some functions as shown in Fig. 3(d) and (e). The reason is that a small frozen number makes the evolutionary speed too fast. It does not have sufficient evolution time for solutions to converge to the desired values, and some important information is lost. Therefore, it cannot find better solutions and premature convergence occurs. In summary, compared to RGA and RSAGA, the proposed algorithm has better performance in finding the global optima, and if the frozen number is properly chosen, its performance could be even better.

3.3. Performance of the proposed algorithm

In this section, the proposed algorithm, ARSAGA, is tested by sixteen benchmark functions as listed in Table 4. These functions have different characteristics: continuity/discontinuity, high dimensionality, large search space, determinism/stochastics, and quadratic/nonquadratic. These functions have 30 dimensions or 100 dimensions such that they may have numerous local minima. Hence, these functions are often utilized to test optimization methods. The first 11 functions are to be minimized, and the others are to be maximized. In these

Table 4
Sixteen Test functions bed

Test functions	Feasible solution space
$f_1(x_i)=\sum_{i=1}^{N}-x_i\sin(\sqrt{\lvert x_i\rvert})$	$[-500,500]^{N=30}$
$f_2(x_i)=\sum_{i=1}^{N}(x_i^2-10\cos(2\pi x_i)+10)$	$[-5.12,5.12]^{N=30}$
$f_3(x_i)=-20\exp\left[-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2}\right]-\exp\left[\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)\right]+20+\exp(1)$	$[-32,32]^{N=30}$
$f_4(x_i)=\frac{1}{4000}\sum_{i=1}^{N}x_i^2-\prod_{i=1}^{N}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$[-600,600]^{N=30}$
$f_5(x_i)=\sum_{i=1}^{N}x_i^2$	$[-100,100]^{N=30}$
$f_6(x_i)=\sum_{i=1}^{N}x_i^4+\text{random}[0,1)$	$[-1.28,1.28]^{N=30}$
$f_7=\sum_{i=1}^{N}\lvert x_i\rvert+\prod_{i=1}^{N}\lvert x_i\rvert$	$[-10,10]^{N=30}$
$f_8(x_i)=\sum_{i=1}^{N}\left(\sum_{j=1}^{i}x_i\right)^2$	$[-100,100]^{N=30}$
$f_9=\min\{\lvert x_i\rvert,\quad i=1,2,\ldots,N\}$	$[-100,100]^{N=30}$
$f_{10}(x_i)=\sum_{i=1}^{N}\text{int}(x_i)$	$[-100,100]^{N=30}$
$f_{11}(x_i)=\sum_{i=1}^{N}(x_i-i)^2$	$[-100,100]^{N=30}$
$f_{12}(x_i)=\sum_{i=1}^{N}x_i^2$	$[-100,100]^{N=30}$
$f_{13}(x_i)=\sum_{i=1}^{N}-x_i\sin(\sqrt{\lvert x_i\rvert})$	$[-500,500]^{N=30}$
$f_{14}=\frac{1}{N}\sum_{i=1}^{N}(x_i^4-16x_i^2+5x_i)$	$[-5,5]^{N=100}$
$f_{15}(x_i)=\sum_{i=1}^{N}(-1)^{i+1}x_i^2$	$[-100,100]^{N=30}$
$f_{16}(x_i)=\sum_{i=1}^{N}\text{int}(x_i)$	$[-100,100]^{N=100}$

functions, f_1–f_9 are adopted from Leung and Wang [16]. In using the proposed algorithm, the maximum generation number is limited to 10000, the initial population size M_1 is set to 200, and the population size M_2 is 20. The other parameters are unchanged as listed in Table 1. The stop criterion is that the searched optimum value is better than the given accept value or the maximum generation number is complete. Under these conditions, 50 independent runs are executed for each function. It is noted that the accept values of f_1–f_9 are obtained from Leung and Wang [16] and the remainder functions have not been studied in their work. Since reasonable frozen numbers are suggested in Section 3.2, three different frozen numbers, $N_{\text{frozen}} = 3$, 10, and 20, will be further discussed in this section.

The optimization histories of these functions are shown in Fig. 4, and their mean function values of 50 runs are shown in Table 4. For all 16 functions, the proposed algorithm with $N_{\text{frozen}} = 3$ always has the best results and the fast convergence speed, while that with $N_{\text{frozen}} = 20$ always has the worst results and the slowest convergence speed. This shows that the proposed algorithm not only makes the convergence speed fast but also has better hill-climbing ability if a suitable frozen number is adopted. Hence, if the adaptive mechanisms are properly triggered, the proposed hybrid algorithm could improve the drawbacks of GA significantly.

The mean numbers of function evaluation and the mean function values by the proposed algorithm with three different frozen numbers are listed in Table 5. The number of function evaluation is the number of function call to evaluate the function value during the execution of the algorithm, and it is often used to represent the execution time of the algorithm. For all 16 functions, the mean number of function evaluation by the proposed algorithm with a smaller frozen number is significantly less than that with a bigger frozen number. In other words, the proposed algorithm with a small frozen number needs less

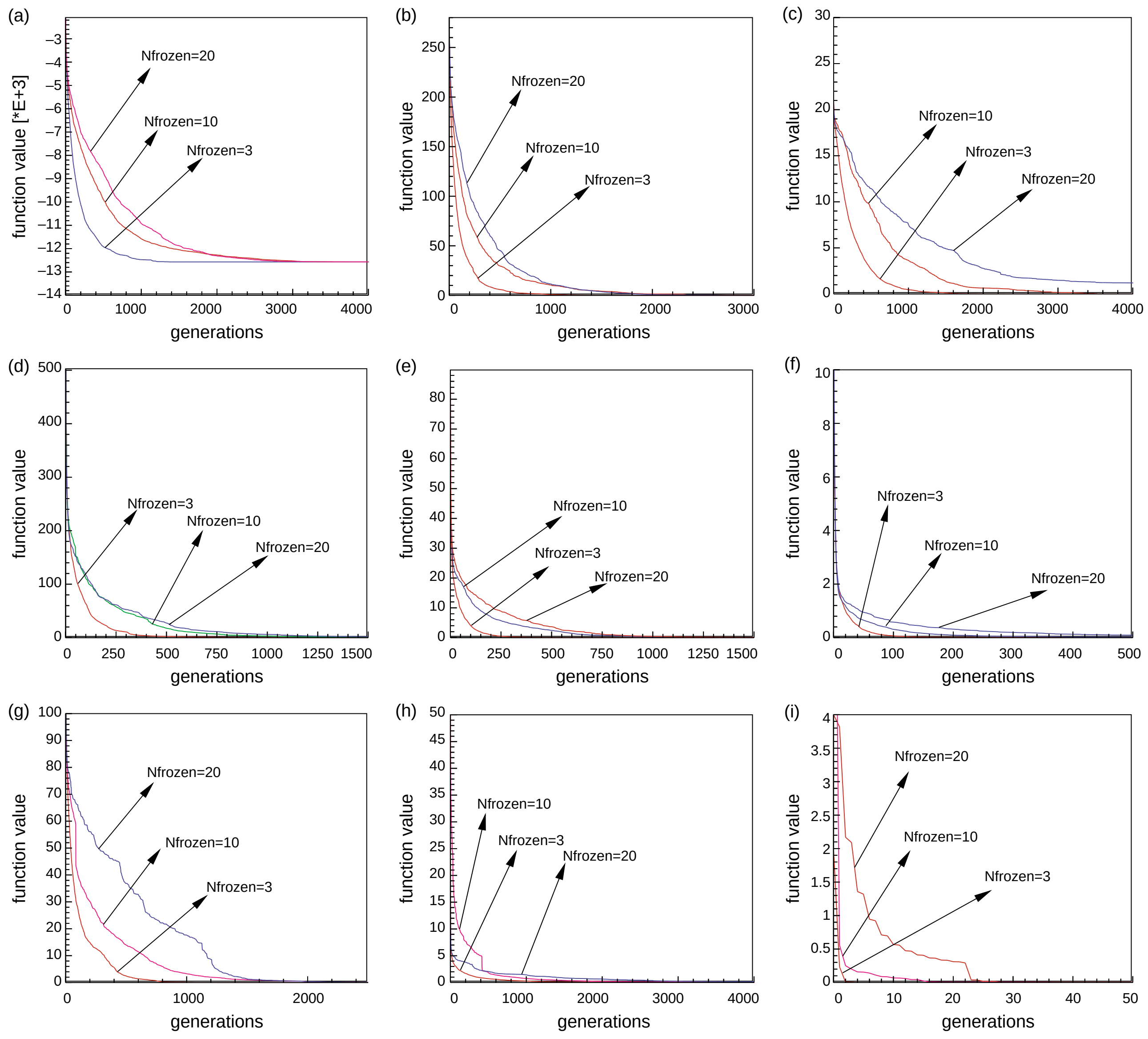

Fig. 4. Convergence of 16 test functions with different frozen number. (a) F_1 (b) F_2 (c) F_3 (d) F_4 (e) F_5 (f) F_6 (g) F_7 (h) F_8 (i) F_9 (j) F_{10} (k) F_{11} (l) F_{12} (m) F_{13} (n) F_{14}(o) F_{15} (p) F_{16}.

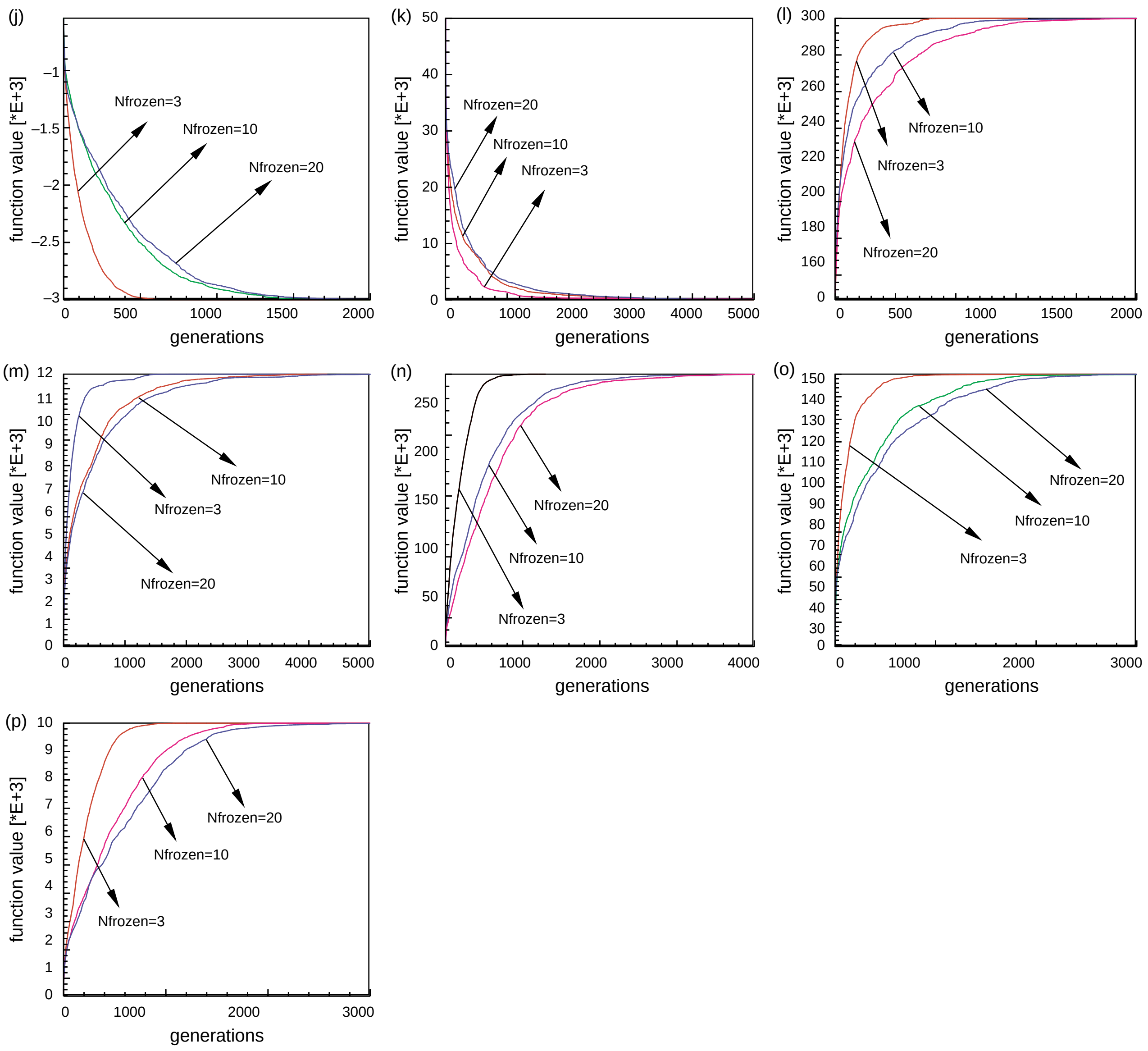

Fig. 4 (*continued*)

computational time and is more efficient. The mean numbers of function evaluation by Leung and Wang for the first nine functions are also listed in the brackets of column 4 of Table 5. Compared to these results, the present algorithm with $N_{\text{frozen}}=3$ has much small mean number of function evaluation except f_8. Although the present algorithm has larger mean number of function evaluation in f_8 than that by Leung and Wang, these two numbers are close. If the frozen number is equal to 20, the present algorithm still could compete with that of Leung and Wang. Among the nine functions, the present algorithm has less mean number of function evaluation in four functions, and the algorithm of Leung and Wang has less mean number of function evaluation in five functions. It should be noted that the algorithm of Leung and Wang (OGA/Q) is complicated. As for the mean function values of 50 simulations, if the frozen number is 20, there are seven functions whose global optimum values could not be found. Among these seven functions, the near global optimum values are found for f_1, f_6, f_{13}, and f_{15}. Hence, by this frozen number, it has difficulties only for f_3, f_8, and f_{11}. If the frozen number is 3, the global optimum values of f_3 and f_8 are found, and the near global optimum values are further improved for f_1, f_6, f_{13}, and f_{15}. The mean function values for the first nine functions obtained by Leung and Wang are also listed in Table 5. As indicated, if the stuck number is 20, the present algorithm has better mean function value than the algorithm of Leung and Wang except for f_3 and f_8. If the stuck number is equal to 3, the present algorithm has better or the same results for these nine functions. If one considers f_1 and f_6 that are difficult to find the global optimum values, the present algorithm has better mean values than the algorithm of Leung and Wang. Also, note that the mean function values by the present algorithm with $N_{\text{frozen}}=3$ are always better than the given accept values or equal to the global optimum values. The standard deviation of the function values by 50 runs is also shown in Table 5. It is

Table 5
Results by ARSAGA with different frozen number

Test function	Mean number of function evaluates		Mean function value (standard deviation)					Global optimum value (accept value)
	3	10	20	3	10	20	Leung and Wang	
f_1	49660	97746	117203 [302166]	−12569.46(0)	−12569.465 (0)	−12569.468 (0)	−12569.4537 (6.447×10^{-3})	−12569.5 (−12569.4537)
f_2	65922	168320	187656 [224710]	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
f_3	112072	277283	399178 [112421]	0 (0)	0 (0)	0.961244 (4.18928)	0 (0)	0 (0)
f_4	121381	316128	360408 [134000]	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
F_5	107380	299152	344262 [112559]	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
f_6	15406	37926	48051 [112652]	0.00473 (0)	0.00482 (0)	0.00496 (0)	0.00630 (4.069×10^{-4})	0 (0.00630)
f_7	48978	275254	365224 [112612]	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
f_8	165780	400200	400200 [112576]	0 (0)	15.68129 (14.98217)	27.19699 (26.34479)	0 (0)	0 (0)
f_9	809	1463	2095 [112893]	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
f_{10}	24948	41134	56398	−3000 (0)	−3000 (0)	−3000 (0)	–	−3000 (−3000)
f_{11}	363979	400200	400200	0.021679 (0.02732)	0.6365412 (0.87274)	0.876560 (1.39688)	–	0 (0)
f_{12}	22024	32405	50141	300000 (0)	300000 (0)	300000 (0)	–	300000 (300000)
f_{13}	65165	107455	131244	12569.481 (0)	12569.482 (0)	12569.483 (0)	–	12569.5 (12569.48)
f_{14}	22640	63631	96360	250 (0)	250 (0)	250 (0)	–	250 (250)
f_{15}	254541	322634	324736	149999.99 (0.06622)	149999.97 (0.13713)	149999.92 (0.54269)	–	150000 (150000)
f_{16}	24774	60274	86535	10000 (0)	10000 (0)	10000 (0)	–	10000 (10000)

evident that a smaller frozen number gives smaller standard deviation than the bigger ones, and it gives a more stable solution. This may indicate that the reliability of the proposed algorithm is very high. Hence, it can be said that the proposed algorithm could find stable values with high precision in less computation time.

3.4. Minimum cost of a welded beam

The initial configuration of this optimization problem is shown in Fig. 5. A beam denoted as A and made by low-carbon steel (C-1010) is to be welded to a rigid support member B. The length L of the welded beam is fixed as 14 in. and it is designed to support a load F of 6000 lb. The width and height of this solid beam is denoted as *b* and *t*, respectively. Besides, the length of welding is *l* and the width of welding is *h*. Hence, the design variables are $\{h, l, t, b\}$ as shown in Fig. 5. The objective function is the cost of the system, and penalty functions are used to transform this constrained problem to an unconstrained problem. Therefore, the objective function of this problem is given as

$$f = (1 + C_1)h^2 l + C_2 tb(\mathrm{L} + l) + \sum_{i=1}^{m} P_i \tag{24}$$

where C_1 is the cost per volume of the welded material, C_2 is the cost per volume of the bar stock, and m is the total number of structural constraints. The ith penalty function P_i is defined as

$$P_i = \begin{cases} 0 \\ M \text{ when the } i\text{th constraint is violated} \end{cases} \tag{25}$$

where M is a big positive constant. This problem is subjected to the following constraints adopted from [28]:

$$\phi_1 = \bar{\tau} - \tau \geq 0 \tag{26}$$

$$\phi_2 = \bar{\sigma} - \sigma \geq 0 \tag{27}$$

$$\phi_3 = b - h \geq 0 \tag{28}$$

$$\phi_4 = l \geq 0 \tag{29}$$

$$\phi_5 = t \geq 0 \tag{30}$$

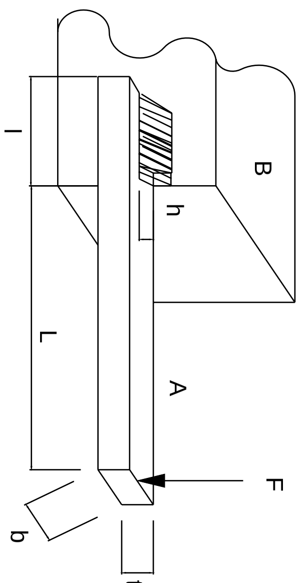

Fig. 5. Initial configuration of a welded beam.

Table 6
Search spaces of design variables and values of specifications

Item	Search space or values
H	[0.125,20.0]
L	[0.1,20.0]
T	[0.1,20.0]
B	[0.1,20.0]
C_1	0.10471(\$/in^3)
C_2	0.04811(\$/in^3)
$\bar{\tau}$	13600 psi
$\bar{\sigma}$	30000 psi
E, Young's modulus of bar stock	30×10^6 psi
G, shear modulus of bar stock	12×10^6 psi

$$\phi_6 = P_b - F \geq 0 \tag{31}$$

$$\phi_7 = h - 0.125 \geq 0 \tag{32}$$

$$\phi_8 = 0.25 - \delta \geq 0 \tag{33}$$

where τ and σ are the maximum shear stress in the welded material and the maximum normal stress in the bar material, respectively. The design shear stress of the welded material and the normal stress of the bar material are denoted as $\bar{\tau}$ and $\bar{\sigma}$, respectively. The buckling load of the bar material is P_b and the end deflection of the bar material is δ. This example needs to not only find the sizes of the beam but also find the sizes of the welding under many structural constraints, and it has studied by several researchers to test the performance of their optimization algorithms [6,29,30]. For this problem, 100 independent simulations are executed, and the test conditions are the same as those in Section 3.3 except that the algorithm stops when the maximum generation number is complete. The search spaces of the design variables and the values of some specifications are listed in Table 6.

Ragsdell and Phillips [29] used five different methods (APPROX, DAVID, GP, SIMPLEX, RANDOM) to investigate this problem, and Deb [30] proposed a canonical GA (CGA) to compare it with these five methods. Leite and Topping [6] compared their method (GEBENOPT) with CGA and these five methods. Among these methods, the method called APPROX used by Ragsdell and Phillips [29] has better performance than the other techniques. Therefore, the result obtained by APPROX, which is equal to 2.38, is considered as the accept value for this optimization problem. The results by the proposed algorithm with different frozen numbers are compared in Table 7. In this table, the number of feasible solutions in 100 independent simulations means the number of the searched solutions that do not violate any constraint. The number of successful hits means the number of the searched solutions that are feasible and better than the accept value. From Table 7, it shows that the proposed algorithm has very high reliability and stability in searched solutions when the frozen number is small such as 3, 5, and 10. Especially, it shows very excellent performance when the frozen number is 3 or 5. Even though it is an engineering optimization problem with many constraints, the proposed method has the same excellent performances as in the previous section. When the frozen number is 20, the performance of the proposed method is poor as compared to other frozen numbers due to large deviation and low reliability. However, its results are still acceptable because the reliability of solution is up to 65% under such strict constraint conditions. The search histories of ARSAGA with different frozen numbers are also shown in Fig. 6. It shows that the proposed algorithm has fast convergence speed to the desired solution under a small frozen

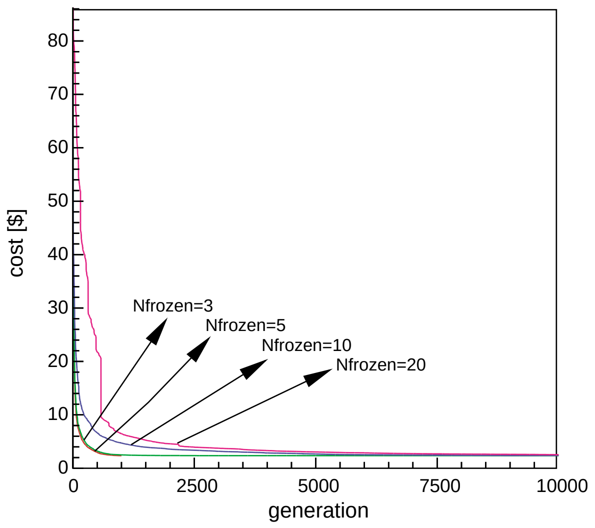

Fig. 6. Average cost histories for ARSAGA with different frozen numbers in 100 simulations.

Table 7
Comparison of results by ARSAGA with different frozen numbers

Item	ARSAGA			
	$N_{\text{frozen}}=3$	$N_{\text{frozen}}=5$	$N_{\text{frozen}}=10$	$N_{\text{frozen}}=20$
Accept cost	2.38	2.38	2.38	2.38
Mean number of function evaluations	26466	42268	178880	296460
Mean cost	2.26	2.26	2.30	2.52
Standard deviation	0.0078	0.0078	0.2082	0.3272
Number of feasible solution	100	100	100	100
Number of successful hits	100	100	94	65

Table 8
Comparison of results by ARSAGA and other algorithms

Methods	Design variables				Cost
	h	l	t	b	
APPROX-best	0.2444	6.2189	8.2915	0.2444	2.38
DAVID-best	0.2434	6.2552	8.2915	0.2444	2.38
GP-best	0.2455	6.1960	8.2730	0.2455	2.39
SIMPLEX-best	0.2792	5.6256	7.7512	0.2796	2.53
RANDOM-best	0.4575	4.7313	5.0853	0.6600	4.12
GEBENOPT-best	0.2489	6.1097	8.2484	0.2485	2.40
CGA-best	0.2489	6.1730	8.1789	0.2533	2.43
ARSAGA-best	0.2231	1.5815	12.8468	0.2245	2.25
ARSAGA-average	–	–	–	–	2.26
ARSAGA-worst	0.1988	1.7977	12.7857	0.2267	2.28

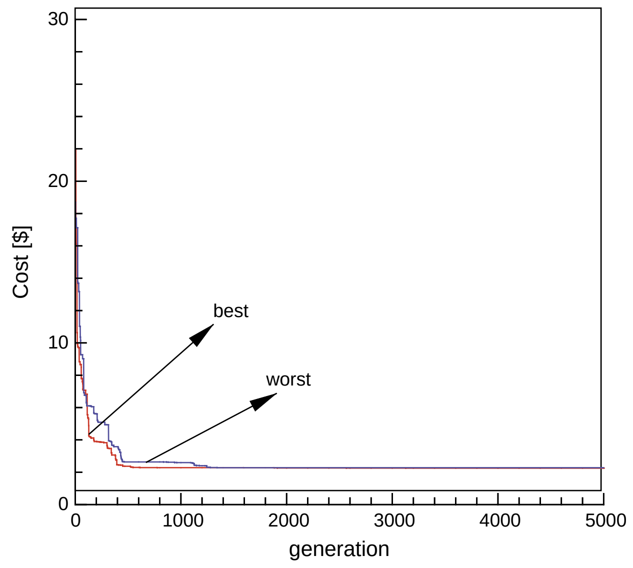

Fig. 7. The cost histories for ARSAGA with $N_{\text{frozen}}=3$ in 100 simulations.

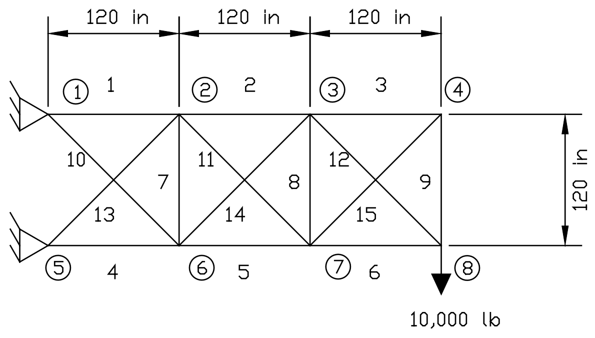

Fig. 8. Initial geometry of 15-bar truss.

number. If the frozen number is in a reasonable range, the smaller the frozen number, the faster the convergence speed.

The results obtained by the proposed method with $N_{\text{frozen}}=3$, the five methods in [29], CGA in [30], and GEBENOPT in [6] are listed in Table 8. The 'best' in the table means that the final searched cost is the best among 100 simulations. Similarly, the 'worst' means that the final searched cost is the worst among 100 simulations. It shows that the proposed method has improved the cost significantly as compared to the other methods. Considering the worst case of the proposed method, its searched cost that is 2.28 is still much better than the values by the other methods. Besides, the convergence histories of the proposed method with $N_{\text{frozen}}=3$ are shown in Fig. 7. It shows that the proposed method not only find a better solution but also has fast convergence speed. Note that it converges to the desired cost at about the 500th generation in the best case and at about the 1100th generation in the worst case.

3.5. Minimizing the weight of a 15-bar truss

This problem is adopted from Kunar et al. [31], in which it was treated as a continuous problem. The initial geometry of this 15-bar truss is shown in Fig. 8, and its cross sectional areas are treated as discrete values under stress constraints. A load of 10,000 lb is applied at joint number 8 of the cantilever truss. The stress of each member is limited to be less than 25,000 psi in both tension and compression for all members. The Young's modulus of the material is 1.0×10^7 psi and its density is 0.1 lb-in^{-3}. This problem is solved in both discrete sizing variables and continuous configuration variables. In the configuration optimization, the x and y coordinates of joint numbers 2, 3, 6, and 7 are allowed to vary. Also, joint numbers 6 and 7 have the same x coordinates as joint numbers 2 and 3, respectively. In other words, joint numbers 6 and 7 are allowed to vary only in y coordinates. Moreover, joint numbers 4 and 8 are fixed in x coordinates, and their y coordinates are allowed to vary. Hence, there are 23 design variables totally, which includes fifteen sizing variables and eight configuration variables, and it could be considered as a complex problem. It is noted that the original point of $x-y$ Cartesian coordinate is set at joint number 5. In this case, the population size M_1 is equal to 80, M_2 40, the maximum generation number 200, and the frozen number 10. The other parameters are unchanged as listed in Table 1. The objective function of this problem is represented as:

$$f(\tilde{x}) = W(\tilde{x}) + \sum_{i=1}^{m} Pf_i \tag{34}$$

where $W(\tilde{x})$ represents the weight of the truss determined by the design variable set $\tilde{x}$, Pf_i is the ith penalty function, and m is the total number of stress constraints. The penalty function Pf_i

Table 9
The feasible search space of design variables of a 15-bar truss

Design variable	Feasible search space
Cross-sectional area: A_1–A_{15}	{0.111, 0.141, 0.174, 0.220, 0.270, 0.287, 0.347, 0.440, 0.539, 0.954, 1.081, 1.174, 1.333, 1.488, 1.764, 2.142, 2.697, 2.800, 3.131, 3.565, 3.813, 4.805, 5.592, 6.572, 7.192, 8.525, 9.300, 10.850, 13.330, 14.290, 17.170, 19.180} (in.^{2}).
Coordinate X_2	{100,140} in.
Coordinate X_3	{220,260} in.
Coordinate Y_2	{100,140} in.
Coordinate Y_3	{100,140} in.
Coordinate Y_4	{50,90} in.
Coordinate Y_6	{−20,20} in.
Coordinate Y_7	{−20,20} in.
Coordinate Y_8	{20,60} in.

Table 10
Comparison of results for a 15-bar truss

Design variables	Value of design variable	
	MBGA	ARSAGA
A_1	1.174	0.954
A_2	0.954	1.081
A_3	0.440	0.440
A_4	1.333	1.174
A_5	0.954	1.488
A_6	0.174	0.270
A_7	0.440	0.270
A_8	0.440	0.347
A_9	1.081	0.220
A_{10}	1.333	0.440
A_{11}	0.174	0.220
A_{12}	0.347	0.440
A_{13}	0.174	0.347
A_{14}	0.347	0.270
A_{15}	0.440	0.220
X_2	123.189	118.346
X_3	231.595	225.209
Y_2	107.189	119.046
Y_3	119.175	105.086
Y_4	60.462	63.375
Y_6	−16.728	−20.0
Y_7	15.565	−20.0
Y_8	36.645	57.722
Weight (lbs)	120.528	104.573

is defined as

$$Pf_i = \begin{cases} 0 \\ M \text{ when the } i\text{th constraint is violated} \end{cases} \tag{35}$$

where M is a big positive constant.

Both the feasible search spaces of discrete sizing variables and continuous configuration variables are listed in Table 9. Both the results obtained by the proposed algorithm and a modified BGA (MBGA) based on a fitness scaling technique proposed by Wu and Chow [32] are listed in Table 10. From this table, it shows that the results obtained by the proposed algorithm are better than those obtained by the modified BGA-based method. The proposed algorithm could have 13.2% less weight than the modified BGA-based method. Hence, the proposed method is efficient. The variation histories of the optimum weight by the present algorithm are shown in Fig. 9. It converges to the searched result at about the 150th generation. Also, the optimized truss configuration of this case is shown in Fig. 10.

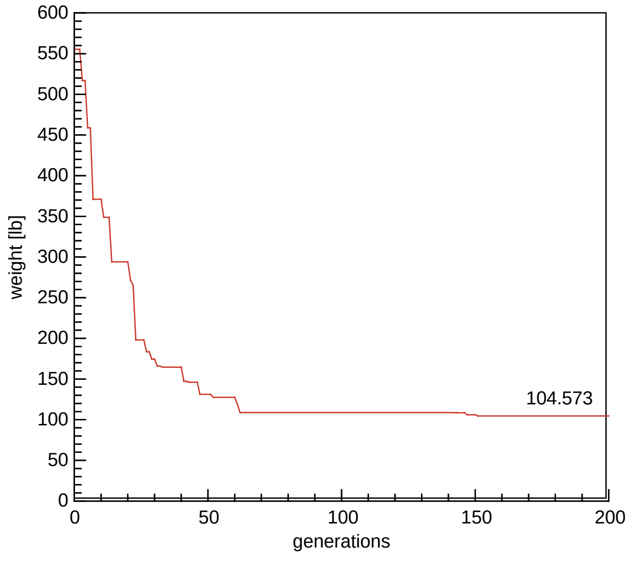

Fig. 9. The optimum weight variation with generation histories.

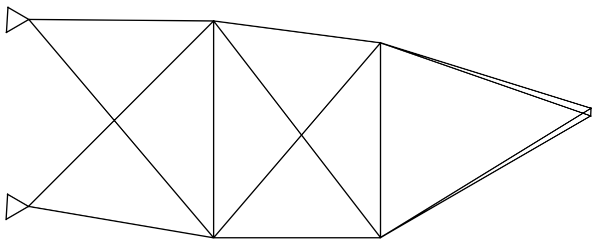

Fig. 10. Optimized configuration of the 15-bar truss.

4. Conclusions

A hybrid stochastic method, named ARSAGA (adaptive real-parameter simulated annealing genetic algorithm), has been proposed. In this algorithm, the merit of crossover operator of GA is maintained, and both SA mutation and GA mutation have been utilized. In addition, adaptive mechanisms are also included. Seven test functions with diverse characteristics are applied to demonstrate the performance of the proposed operators and to compare the performance of the proposed algorithm with other GAs. It shows that the proposed operators out-perform the others. Moreover, the proposed hybrid algorithm is applied to 16 benchmark optimization functions. The results indicate that in a reasonable range of frozen number, the smaller the frozen number, the better the performance of the proposed hybrid algorithm. For all these functions, the proposed algorithm with proper frozen number could find the global optimum values or the near global optimum values. It's also shown that the proposed algorithm is effective and its convergence speed is fast although the functions are complex. Furthermore, the proposed algorithm has nice hill-climbing ability, and it is robust to search space because it could find a solution in reasonable precision with high reliability no matter if the function dimension is 30 or 100. One engineering optimization problem of a welded beam is used to test the proposed technique's applicability. Under strict conditions, the proposed method still has high reliability as well as stability on the searched solutions and fast convergence speed. The second engineering optimization problem is to minimize the weight of a 15-bar truss with stress constraints under both sizing optimization and configuration optimization. This problem has 23 design variables including discrete variables and continuous variables. Under such conditions, the proposed method still has good searched results and fast convergence speed. Hence, the proposed method not only has excellent performance in function optimization but also has high applicability in engineering optimization problems.

References

[1] Dumitrache I, Buiu C. Genetic learning of fuzzy controllers. Math Comput Simul 1999;49:13–26.

[2] Jeong IK, Lee JJ. Adaptive simulated annealing genetic algorithm for system identification. Eng Appl Artif Intell 1996;9:523–32.

[3] Kristinsson K, Dumont GA. System identification and control using genetic algorithm. IEEE Trans Syst Man Cybern 1992;22(5):1033–46.

[4] Huang HC, Pan JS, Lu ZM, Sun SH, Hang HM. Vector quantization based on genetic simulated annealing. Signal Process 2001;81:1513–23.

[5] Soltani AR, Tawfik H, Goulermas JY, Fernado T. Path planning in construction sites: performance evaluation of the dijkstra, A^*, and GA search algorithms. Adv Eng Inform 2002;16:291–303.

[6] Leite JPB, Topping BVT. Improved genetic operators for structural engineering optimization. Adv Eng Soft 1998;29(7):529–62.

[7] Hasançebi O, Erbatur F. Evaluation of crossover techniques in genetic algorithm based optimization structural design. J Comput Struct 2000;78: 435–48.

[8] Griffiths DR, Miles JC. Determining the optimal cross-section of beams. Adv Eng Inform 2003;17:59–76.

[9] Hsieh CH, Chou JH, Wu YJ. Taguchi—MHGA method for optimizing Takagi–Sugeno fuzzy gain-scheduler. In: proceedings of the 2000 automatic control conference Taipei, Taiwan; 2000 p. 523–28.

[10] Nanakorn P, Meesomklin K. An adaptive function in genetic algorithms for structural design optimization. J Comput Struct 2001;79:2527–39.

[11] Wong SV, Hamouda AMS. Optimization of fuzzy rules design using genetic algorithm. Adv Eng Soft 2000;31:251–62.

[12] Adler D. Genetic algorithm and simulated annealing: a marriage proposal. IEEE Int Conf Neural Netw 1993;1104–9.

[13] Tan KC, Li Y, Murray-Smith DJ, Sharman KC. System identification and linearization using genetic algorithms with simulated annealing. Proceedings of the IEEE genetic algorithms in engineering system: innovations and applications, conference publication. vol. 414; 1995. p. 164–9.

[14] Brown D, Huntley C, Spillane AA. A parallel genetic heuristics for the quadratic assignment problem. Proceeding of the third international conference on neural networks. 1989. p. 406–15.

[15] Shapiro AF. The merging of neural networks, fuzzy logic, and genetic algorithms. Insur Math Econ 2002;31:115–31.

[16] Leung YW, Wang Y. An orthogonal genetic algorithm with quantization for global numerical optimization. IEEE Trans Evol Comput 2001;(5): 575–82.

[17] Angelov P. Supplementary crossover operator for genetic algorithms based on the center-of-gravity paradigm. Control Cybern 2001;(30): 159–76.

[18] Chellapilla K. Combining mutation operators in evolutionary programming. IEEE Trans Evol Comput 1998;(2):91–6.

[19] Gen M, Cheng R. Genetic algorithms and engineering design. New York: Wiley; 1997.

[20] Yen J, Lee B. A simplex genetic algorithm hybrid. In: Proceedings of the IEEE international conference on evol comput Indianapolis, IN; 1997. p. 175–80.

[21] Dejong KA. Analysis of the behavior of a class of genetic adaptive systems. PhD Thesis. University of Michigan, Ann Arbor, MI; 1975.

[22] Grefenstette JJ. Optimization of control parameters for genetic algorithms. IEEE Trans Syst Man Cybern 1986;SMC-16(1):122–8.

[23] Schaffer JD, Caruana RA, Eshelman LJ, Das R. A study of control parameters affecting online performance on genetic algorithms for function optimization. Proceeding of the third international conference on genetic algorithms. Los Altos, CA: Morgan Kaufman; 1989 p. 51–60.

[24] Tsallis C, Starolo DA. Generalized simulated annealing. Physica A 1996; 233:395–406.

[25] Fogel DB. An introduction to simulated evolutionary optimization. IEEE Trans Neural Netw 1994;5(1):3–14.

[26] Huang YP, Huang CH. Real-valued genetic algorithms for fuzzy grey prediction system. Fuzzy Set Syst 1997;87:265–76.

[27] Schwefel HP. Numerical optimization of computer models. New York: Wiley; 1981.

[28] Reklaitis GV, Ravindran A, Ragsdell KM. Engineering optimization methods and applications. New York: Wiley; 1983.

[29] Ragsdell KM, Phillips DT. Optimal design of a class of welded structures using geometric programming. ASME J Eng Ind Ser B 1976;98(3): 1021–5.

[30] Deb K. Optimal design of a welded beam via genetic algorithms. AIAA J 1991;29(11):2013–5.

[31] Kunar RR, Chan ASL. A method for the configurational optimization of structures. Comput Methods Appl Mech Eng 1976; 7(3):331–50.

[32] Wu SJ, Chow PT. Integrated discrete and configuration optimization of trusses using genetic algorithms. J Comput Struct 1995;55(4): 695–702.