ELSEVIER

Advances in Engineering Software 37 (2006) 406–418

ADVANCES IN ENGINEERING SOFTWARE

www.elsevier.com/locate/advengsoft

Improving real-parameter genetic algorithm with simulated annealing for engineering problems

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

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

Received 9 February 2005; received in revised form 2 August 2005; accepted 23 August 2005
Available online 17 October 2005

Abstract

With its inheritance, genetic algorithm may spend much computation time in the encoding and decoding processes. Also, since genetic algorithm lacks hill-climbing capacity, it easily falls in a trap and finds 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 added to insure the solution quality and to improve the convergence speed. The performance of this proposed algorithm is demonstrated in some test functions and two examples. The first example is a helical spring optimization design case, and the second example is a system identification problem described by an ARMAX (auto regressive and moving average exogenous) model. The former is a constrained engineering optimization problem, and the latter is an unconstrained one. The results indicate that the global searching ability and convergence speed of this novel hybrid algorithm is significantly improved, even though small population size is used for a complex and large problem. The proposed algorithm is significantly better than the other genetic algorithm-based methods or other methods discussed in this paper.

Keywords: Genetic algorithm; Simulated annealing; Adaptive mechanism; System identification; Optimization design

1. Introduction

Genetic algorithm (GA) developed by Holland is based on the Darwinian theory of biological evolution [1,2]. It is a very important stochastic search algorithm for solving optimization problems during the last two decades. GA has been widely and successfully applied to various problems like operation research, image processing, and control problems [3–5], etc. GA is different from most conventional calculus-based search algorithms in several ways. GA makes no limitation on the search space of optimization problems. Also, GA searches for the optimum solutions through a population of solutions not just a single solution. These make the GA-based methods more possible to obtain the optimum solutions or near optimum solutions. However, GA codes the parameter sets of an optimization problem as finite-length strings. Instead of the parameters themselves, these strings are further manipulated. Hence, GA may spend much computation time in the encoding and decoding processes. During these processes, some important genes may be lost and the accuracy of the searched solution may also be lost. In the real world, most problems to be solved are complex, and the space of the parameters to be found is continuous not discrete. Therefore, the bits of the parameters will be large if high precision of the parameter values is desired. It will not only waste much computation time but also occupy large computer memory. Besides, since GA deeply depends on the crossover process, it is not easy for GA to find the desired solutions such that its convergence speed is slow. Moreover, GA's ability of fine-tuning solution is not good, because the selected mutation probability may be too small to escape a trap. Anyway, GA is an important but not powerful search method.

Simulated annealing (SA) developed by Metropolis is another important stochastic adaptive method based on the physical process of annealing. In this algorithm, a solid is heated up to a liquid phase by increasing temperature and followed by slowly lowering the temperature for cooling. In this way, it takes up a configuration in a low energy ground state [6]. SA is very powerful in solving complicated problems such as combinational problems. Simulated annealing is also good at hill climbing for the optimum solutions. However, due to the random processes to search for the minimum energy

* Corresponding author. Tel.: +886 5 534 2601x4143; fax: +886 5 531 2062.
E-mail address: hwangsf@yuntech.edu.tw (S.-F. Hwang).

0965-9978/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2005.08.002

state, the convergence speed of simulated annealing is very slow [7].

To avoid the premature convergence and improve the hill-climbing ability of GA, Haupt and Haupt [8] defined a new genetic operator. Adler [9] changed the crossover probability and mutation probability, and Wong and Hamouda [10] used a strong initial population. Another possibility is to merge GA with other powerful methods, such as SA. For example, the standard crossover and mutation processes of GA were replaced by SA operators in Brown et al. [11]. This combined algorithm can keep the advantages and avoid the disadvantages of both search algorithms. Owing to the existence of SA, this combined algorithm has better fine-tuning ability to search for the global optimum solutions and more strong hill-climbing ability for escaping from local minima than the standard GA [7,12].

This paper proposes a novel hybrid algorithm, adaptive real-parameter simulated annealing genetic algorithm (ARSAGA), which is based on a real-parameter genetic algorithm (RGA) with novel operators from SA and adaptive mechanisms. It is designed to improve the fine-tuning ability, the hill-climbing ability, and the speed of convergence to the global optimum solutions or near global optimum solutions. Besides, it should be robust to the search space. The proposed novel hybrid algorithm is presented in Section 2. Section 3 shows the performance of ARSAGA in various engineering problems including identifying a dynamic system represented by an ARMAX (auto regressive and moving average exogenous) model and volume optimization design of a helical spring. The former is an unconstraint optimization problem, and the latter is a constrained optimization problem. Both the constrained problem and unconstrained problem are utilized to test the proposed method. Finally, conclusions are given in Section 4.

2. Adaptive real-parameter simulated annealing genetic algorithm

2.1. Real-parameter genetic algorithm

One major drawback of a binary genetic algorithm (BGA) is that it encodes parameters as finite-length strings such that much computation time is wasted in the encoding and decoding processes. A real-parameter genetic algorithm (RGA) is proposed to overcome this problem. Instead of the coding processes, RGA directly handles the parameters themselves and much computation time is saved. As for the main genetic operators, RGA is the same as BGA in the reproduction process, but they are 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 solution's fitness and uses the well-known roulette selection criterion to create the next population. The solutions with higher normalized fitness have a higher survival probability into the next generation.

In the crossover process of RGA, two solutions called parents are randomly selected from a mating pool. The parents are to do the crossover process if the crossover probability P_c is larger than a random value between 0 and 1. Then the genetic operation of sexual recombination is applied to the pairs of parameters of the parents. Since the parameter pairs to do the crossover process are randomly selected, how to obtain the new parameters without overflowing in the domain and with maintaining the merit of the crossover process is a problem. In general, there are three different genetic operation modes of sexual recombination: (1) one-point crossover, (2) two-point crossover, and (3) uniform crossover [13,14].

The one-point crossover denoted as mode 1 in this paper has only one pair of parameters to do the crossover process. Suppose that the parents could be described as $y_1=\{a_1,a_2,\ldots,a_n\}$ and $y_1=\{b_1,b_2,\ldots,b_n\}$ where the parameters a and b are real numbers and n is a positive integer representing the number of the parameters to be searched. 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

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

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

Here, α_i is a random number and $0 \leqq \alpha_i \leqq 1$. The two-point crossover is referred to as mode 2. If the crossover points are i and j all the parameter pairs between i and j have to do the crossover 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}'), …, and (a_j', b_j') are similar to Eqs. (3) and (4). The uniform crossover is called mode 3 in this paper. The crossover process is applied to every pair (a_i,b_i) of the parents. The expressions of the new pairs are similar to Eqs. (3) and (4).

In addition, a novel crossover mode is proposed here for improving the performance of the crossover process. In this mode, a random crossover point is generated and the crossover process is done from this selected point to either the end point or the start point of the solution, which is randomly selected. For example, if the generated number is i and the end point 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 $j=i\sim n$, are similar to Eqs. (3) and (4).

Finally, the mutation operation is introduced. In this operation, the uniform mutation mode is adopted. Each parameter in a solution is to be mutated or not according to the mutation probability P_m. For example, after the previous

process of crossover, the parents are represented as y'_1 and y'_2, respectively. When a parameter point i in a parent is selected to be mutated, the new parameter value after mutation is

$$a'_i = \alpha_i a'_{i,\mathrm{L}} + (1-\alpha_i)a'_{i,\mathrm{U}} \tag{9}$$

where α_i is a random number, $0 \leqq \alpha_i \leqq 1$, and $a'_{i,\mathrm{L}}$ and $a'_{i,\mathrm{U}}$ are the lower bound and upper bound of the parameter a_i, respectively.

2.2. Simulated annealing

SA is the simulation of annealing of physical many-particle system for finding the global optimum solutions of a large combinatorial optimization problem. The fundament of statistical physics and optimization of combinatorial problems 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 $\mathrm{Pro}_{(\mathrm{Eq})}$ with the energy E_q in a state q as

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

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 at time t, then a new configuration r of the system at time $t+1$ is generated randomly. The configuration r is accepted according to

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

If $E_r - E_q < 0$, then configuration r is accepted as the next configuration at time $t+1$. If $E_r - E_q \geq 0$, the probability of acceptance of this new configuration is decided by the acceptance probability $\mathrm{Pro}_{(r)}$. 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

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

where $c_{(q)}$ is the cost function, Cs_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 for 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 cost function $c_{(r)}$ 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 then the 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 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 goes to infinity, $C(t)$ approaches to zero.

When the annealing schedule is applied to an optimization problem, the energy function becomes the objective function, and the configuration of the internal state variables becomes the configuration of parameters of the optimization problem. Some basic steps are needed in the application of the annealing schedule to an optimization problem. First, the cooling schedule such as Eq. (14) should be carefully designed that it is sufficient to converge to the global minimum. Secondly, the new configuration solutions are generated randomly and they are accepted according to the Metropolis criterion. Thirdly, a stop criterion based on the improvement in the energy function to terminate the execution of the algorithm should be carefully chosen.

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. 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 the probability Eq. (13) is converted to Eq. (16)

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

$$\mathrm{Pro}_{(l)} = \mathrm{e}^{-(E_r-E_q)/(T_{(l)})} \tag{16}$$

where l denotes an integer time step, T_0 is an initial constant temperature, $T_{(l)}$ 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

$$\mathrm{Pro}_{(l)} = \Gamma^{-(E_r-E_q)/T_0} \tag{17}$$

The new mutation operator operates as follows. Generate randomly a new solution from the neighborhood of the original solution, which is obtained after the mutation process of RGA. If the value of the new objective function is less than that of the original objective function, that is $E_r - E_q < 0$, the new solution is better than the old one and it is accepted. On the other hand, if $E_r - E_q \geq 0$, the new one is accepted only when its acceptance probability $\mathrm{Pro}_{(l)}$ given in Eq. (17) is larger than a random value 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 is a function of $f_{\max}$ and $f_{\min}$, which represents the maximum and minimum objective function values of the initial population, respectively. In this paper, T_0 can be chosen as

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

where the influence of $f_{\max}$ is excluded. In addition, since a better initial population will lead to a faster convergence to the desired solution, the tournament criterion is applied in the present approach to obtain a better initial population. Two populations of solutions are randomly generated. Then, one solution is randomly taken from each population of solutions, and the solution that has a higher fitness value can become the solution of the initial population.

In searching the optimum solutions, when the best solution keeps the same for some successive generations, the executed algorithm may be stuck at a local minimum, and some changes should be done on the searching strategy of the algorithm. Therefore, adaptive mechanisms are proposed to do the change. In these mechanisms, if the best solution is the same for the lasting N generations and $N > N_{\mathrm{frozen}}$, the crossover probability and mutation probability are changed according to the following two equations.

$$P_{\mathrm{c}} = P_{\mathrm{c0}} + \frac{N - N_{\mathrm{frozen}}}{N}(\alpha - P_{\mathrm{c0}}) \tag{19}$$

$$P_{\mathrm{m}} = P_{\mathrm{m0}} + \frac{N - N_{\mathrm{frozen}}}{N}(\beta - P_{\mathrm{m0}}) \tag{20}$$

where N_{frozen} is a positive integer constant, P_{c0} and P_{m0} are the initial crossover probability and initial mutation probability, respectively. Besides, α and β are constant real numbers. If $N \leq N_{\mathrm{frozen}}$ or the best solution changes such that N is reset to zero, the crossover probability and mutation probability remain equal to their initial values

$$P_{\mathrm{c}} = P_{\mathrm{c0}} \tag{21}$$

$$P_{\mathrm{m}} = P_{\mathrm{m0}} \tag{22}$$

Eqs. (19) and (21) are called adaptive crossover mechanism, and Eqs. (20) and (22) are called adaptive mutation mechanism. If there is no adaptive mechanism included in the algorithm, for example BGA or RGA, the crossover probability and mutation probability will remain the same as their initial values as Eqs. (21) and (22). 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. In ARSAGA, both the adaptive crossover mechanism and the adaptive mutation mechanism are included. The flow chart of ARSAGA is shown in Fig. 1 and is described as follows.

Step 1: Set parameters.
Step 2: Generate two populations with the same population size M_1. Use tournament criterion to select the initial population with population size M_1.
Step 3: Calculate the objective function value and find the fitness of each solution.
Step 4: Apply reproduction process and select the solutions as parents by the roulette wheel criterion.
Step 5: Apply crossover process.
Step 6: Apply mutation process.
Step 7: Apply SA process. After this process, the created population is treated as offspring.
Step 8: Calculate the objective function value and find the fitness of each offspring.

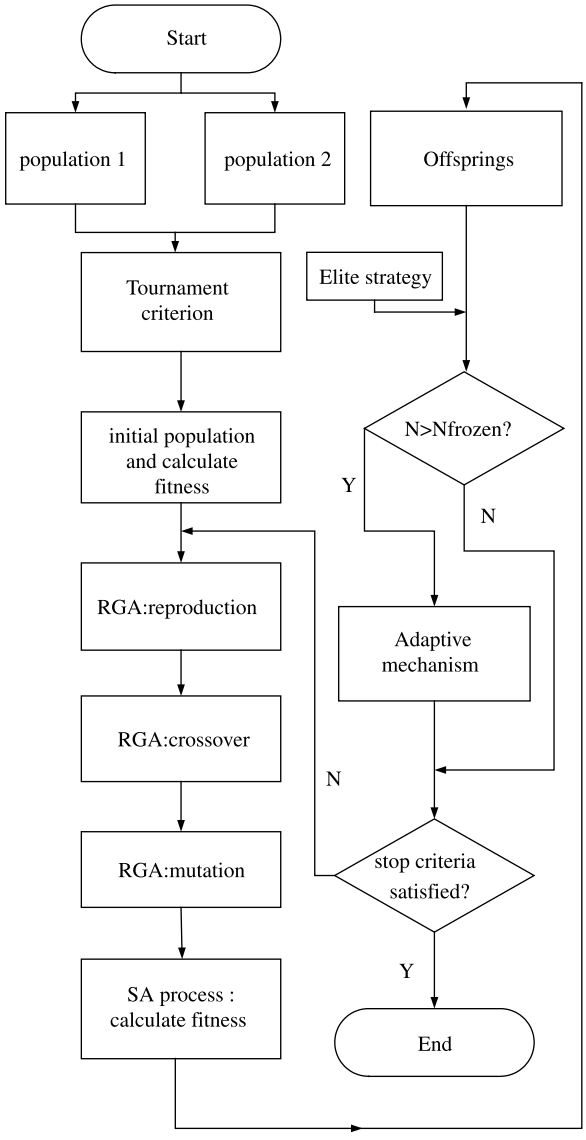

Fig. 1. Flow chart of ARSAGA.

Table 1
Parameter values used in ARSAGA

Parameter	Value
Maximum generation number	600
Population size, M_1	40
Initial crossover probability, P_{c0}	0.7
Initial mutation probability, P_{m0}	0.01
Constant α	0.9
Constant β	0.2

Step 9: If the best solution is not improved, the elitist strategy is adopted to insure the solution quality.
Step 10: If the best solution is the same for N generations and $N > N_{frozen}$, then adaptive mechanisms start.
Step 11: When the executed generation number is equal to the maximum generation number, the algorithm ends.

2.4. Parameter values of ARSAGA

In the proposed algorithm, the values of some parameters need to be selected appropriately. In general, GA-based methods always set population size and maximum generation number to fixed numbers. If a larger population size is used, it has higher probability to obtain better results. But, it takes much computation time and reduces the efficiency of the algorithm. As for the maximum generation number, it will be easy to judge the efficiency of a GA-based algorithm with a large maximum generation number, especially if the algorithm is not efficient. The suitable ranges of crossover probability P_c and mutation probability P_m have been suggested in some articles [15–17]. It is noted that crossover probability P_c is always much larger than mutation probability P_m. Based on the suggested values and some trials, the suitable range of crossover probability P_c is 0.7–0.9 and the suitable range of mutation probability P_m is 0.01–0.2. During the adaptive mechanisms in this work, the values of these two parameters should be limited in their suitable ranges. Hence, the initial crossover probability P_{c0} is equal to 0.7 and constant α is equal to 0.9. Similarly, the initial mutation probability P_{m0} is set to 0.01 and constant β is 0.2. Under the above discussion and some tests, the values of these parameters are listed in Table 1, and they will be adopted in the examples of the following sections.

2.5. Performance of crossover modes

In this section, three functions are utilized to test the capability of the proposed operators in GA-based methods.

Table 2
Test functions

Function number	Function	Feasible solution space
F_1	$f(x_i) = \sum_{i=1}^{5} \text{integer}(x_i)$	$-5.12 \le x_i \le 5.12$
F_2	$f(x_i) = \sum_{i=1}^{10} -x_i \sin\sqrt{\lvert x_i \rvert}$	$5.12 \le x_i \le 5.12$
F_3	$f(x_i) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$	$-2.048 \le x_i \le 2.048$

These functions selected from separated articles [15,18,19] are listed in Table 2. The first two functions are to be minimized and the remainder is to be maximized. In addition to the three common modes of crossover operation, the proposed crossover mode that is referred to as mode 4 is applied to the above functions to compare their efficiency on function optimization. Fig. 2 shows the average and the best values of objective functions in 10 independent simulations by plain RGA with these four different crossover modes. All the parameters are the same as Table 1 except that the maximum generation number is changed to 500 in this and next sections. Besides, the stuck number is not used in this section. The average value is

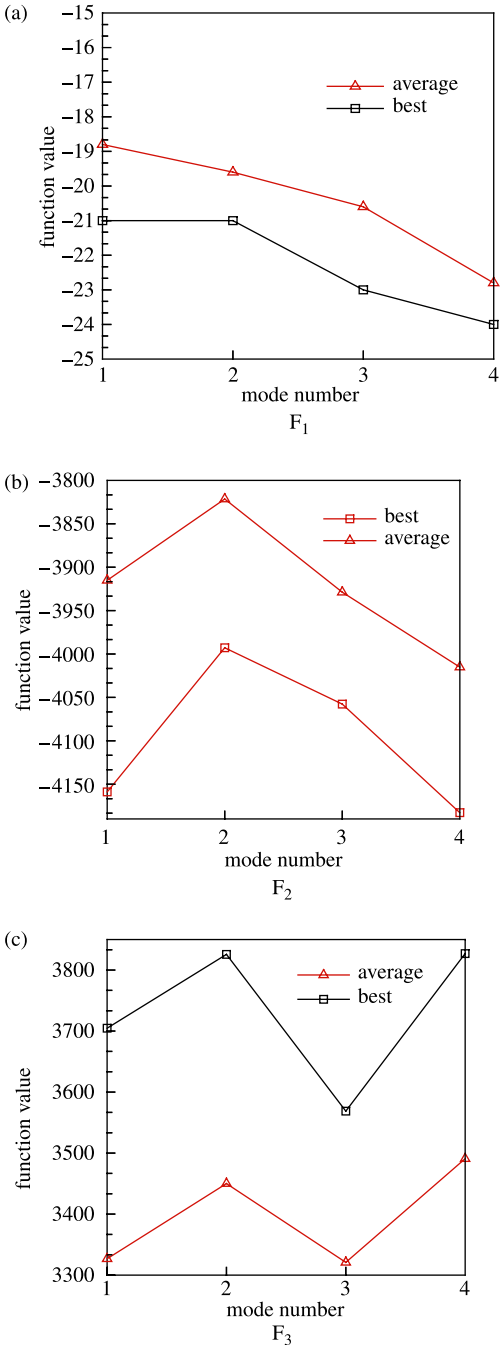

Fig. 2. Results among 10 simulations by RGA with different crossover modes.

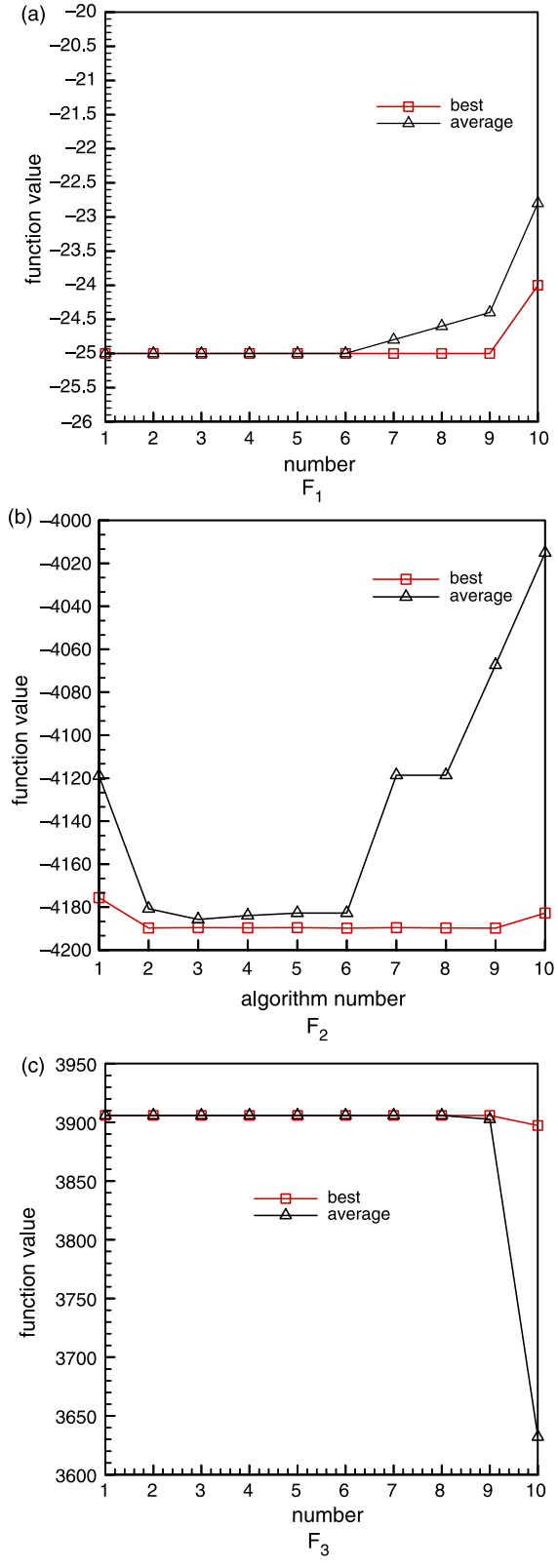

Fig. 3. Function values among 10 simulations by different GA-based methods.

obtained from 10 independent simulations, and the best value means the smallest value of objective function among the 10 independent simulations. Note that the mode number in the abscissa of Fig. 2 represents the corresponding crossover mode.

Table 3
Comparison of function evaluation number by RGA and ARSAGA

Item	F_1 (global value = −25)		F_3 (global value = 3905.9)	
	Convergence generation number	Number of function evaluations	Convergence generation number	Number of function evaluations
RGA-best	>500	>20,000	>500	>20,000
RGA-worst	>500	>20,000	>500	>20,000
N_{frozen} = 30-worst	88	7040	58	4640
N_{frozen} = 30-best	33	2640	3	240
N_{frozen} = 20-worst	79	6320	17	1360
N_{frozen} = 20-best	30	2400	2	160
N_{frozen} = 10-worst	33	2640	8	640
N_{frozen} = 10-best	25	2000	2	160

From the results of the best values, it indicates 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 can 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 by the other three modes. This may imply that the solution found by the proposed crossover mode is more reliable than that by 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 mode increases the diversity of solution such that it can escape from a trap to other search spaces. For this reason, it can search a better solution than the other three modes.

2.6. Performance of adaptive mechanisms and SA

Numbers 1–8 in the abscissa of Fig. 3 represent that the function value is obtained by the proposed ARSAGA with N_{frozen} = 1, 2, 5, 10, 20, 30, 50, and 100, respectively. Number 9 denotes that the results are found by the combination of plain

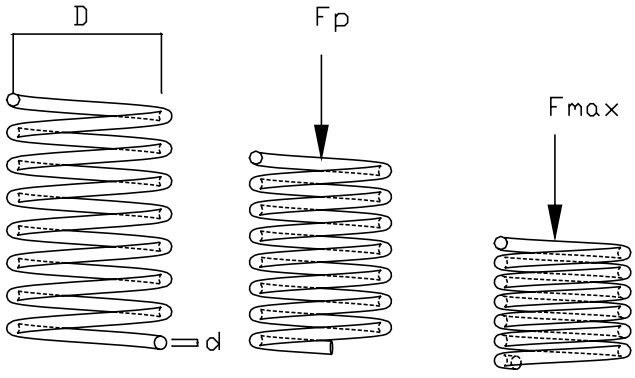

Fig. 4. Configurations of helical spring.

Table 4
Values of some specifications for helical spring design

Input specifications	Value
Allowable stress, S	189,000 psi
Young's modulus, E	0.3×10^8 psi
Shear modulus, G	0.115×10^8 psi
Maximum working load, F_{max}	1000.0 lb
Minimum wire diameter, d_{min}	0.2 in.
Maximum free length, L_{max}	14.0 in.
Maximum outside diameter, D_{max}	3.0 in.
Preload compression force, F_p	300.0 lb
Maximum deflection under preload, δ_{pm}	6.0 in.
Deflection from preload to maximum load, δ_w	1.25 in.
End coefficient, C_E	1.0

RGA with SA. That is to say, compared to ARSAGA, the method of number 9 does not include adaptive mechanisms. For comparison, number 10 is just the plain RGA without including both SA and adaptive mechanisms. From the results of numbers 6–10 in Fig. 3, the plain RGA has always the worst function values, and number 9, that is, also called RSAGA in this work has better results than RGA. This should be attributed to the assistance of SA. Under the assistance of adaptive mechanisms, number 8 could find better solutions than number 9, even though its stuck number N_{frozen} is equal to 100. Among numbers 6–8, the function values are improved as the stuck number is decreased. For numbers 3–6, it looks like there is a stable stuck number from 5 to 30, by which the found function values are almost the same. Considering numbers 1 and 2 where the stuck number is 1 and 2, their results become worse. For example, consider F_2 whose acceptable value is -4180.0.

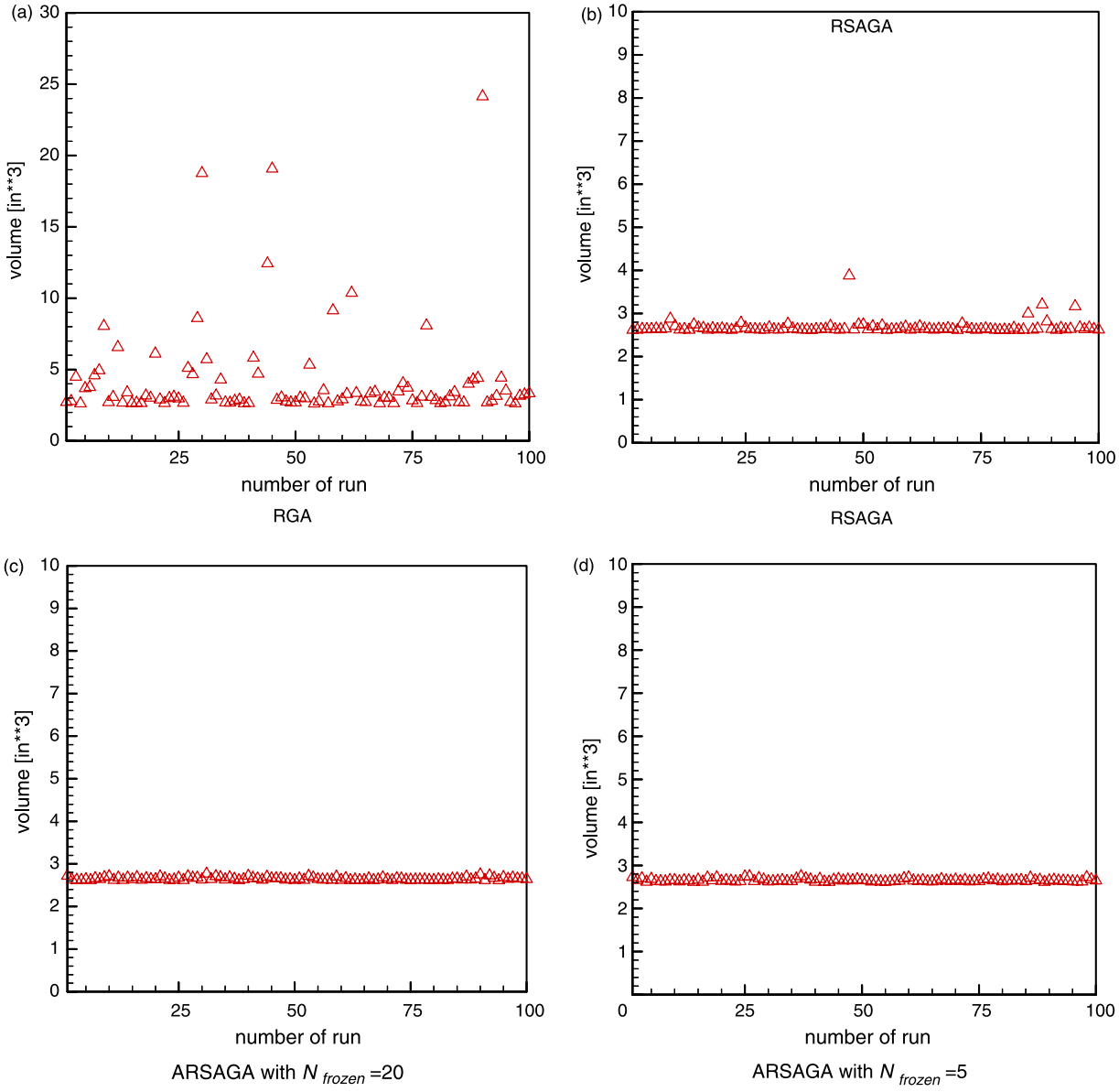

Fig. 5. Searched results of 100 independent simulations for case 1 helical spring.

Table 5
Comparison of the results by different methods for case 1 helical spring

Item	RGA	RSAGA	ARSAGA with N_{frozen}				
			50	30	20	10	5
Average	4.21	2.68	2.66	2.65	2.65	2.65	2.65
Standard deviation	3.458	0.153	0.083	0.045	0.033	0.031	0.029
Number of feasible solutions	100	100	100	100	100	100	100
Accept value [20]	2.84	2.84	2.84	2.84	2.84	2.84	2.84
Number of successful hits	35	96	99	99	100	100	100

From Fig. 3(b), the best objective function value among 10 independent simulations by most algorithms is less than −4189.0, and the average objective function value is less than the acceptable value only when the stuck number is from 5 to 30. That is to say, one could obtain a reliable solution, only if ARSAGA with suitable stuck number is used. As for some simple functions like F_3, whether the stuck 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. Furthermore, the addition of adaptive mechanisms could further improve the function values.

Total numbers of function evaluations of F_1 and F_3 by ARSAGA and RGA are listed in Table 3. The convergence generation number means that the function value is equal to the global value after the executed generation number. Since there are 40 function evaluations in one executed generation of RGA, its total number of function evaluations is equal to 40 times convergence generation number. In the similar way, the total number of function evaluations of ARSAGA is equal to 80 times convergence generation number. From Fig. 3, RGA does not converge to the global value of F_1 and F_3, and it means that the convergence generation number should exceed the maximum generation number (=500). Therefore, the total number of function evaluations of RGA should exceed 20,000 in these test cases. From Table 3, if one consider the worst situation among 10 independent simulations by the proposed method with $N_{frozen}=30$, the convergence generation number for F_1 is 88, and the total number of function evaluations is 7040. Compared with RGA, the computation time of the proposed method has been much reduced. If one consider the best situation among 10 independent simulations by the proposed method with $N_{frozen}=10$, the convergence generation number is 25, and the total number of function evaluations is 2000. Therefore, if an appropriate stuck number is adopted, the proposed method can further reduce the computation time. For test function F_3, the similar results can be concluded. Hence, the proposed method is also efficient in computation time.

3. Verification examples

3.1. Design of a helical spring

This optimization problem is adopted from Siddall [20]. The initial helical spring configurations are shown in Fig. 4. The purpose of this problem is to design the helical spring to have minimum volume under static loading. The design variables are the wire diameter d, the mean coil diameter D, and the active coil number N. It is noted that all these design variables were treated as real numbers by Siddall. The values of some specifications are listed in Table 4. Penalty functions are adopted to transform a constrained problem to an unconstrained problem. Then the objective function of this problem is given as follows

$$\text{volume} = \frac{\pi^2}{4} D d^2 (N + 2C_E) + \sum_{i=1}^{m} \lambda_i \tag{23}$$

where λ_i is the ith penalty function, and m is the total number of structural constraints. The penalty function λ_i is defined as

$$\lambda_i = \begin{cases} 0 \\ M \quad \text{when the } i\text{th constraint is violated} \end{cases} \tag{24}$$

where M is a big positive constant and equals 10^5 in this case. This problem is subjected to the following constraints adopted from [20].

(1) Stress constraint

$$\phi_1 = S - \tau \geq 0 \tag{25}$$

(2) Configuration constraints

$$\phi_2 = L_{max} - l_f \geq 0 \tag{26}$$

Table 6
Values of design variables by different methods for case 1 helical spring

Item	Wire diameter, d (in.)	Coil mean diameter, D (in.)	Number of coils, N	Volume of wire, (in^3)
Reference [20]	0.263	0.91	16.2	2.84
RGA	0.290	1.340	7.458	2.615
RSAGA	0.291	1.367	7.173	2.614
ARSAGA $N_{frozen}=50$	0.289	1.338	7.483	2.616
ARSAGA $N_{frozen}=30$	0.290	1.357	7.275	2.615
ARSAGA $N_{frozen}=20$	0.293	1.413	6.724	2.615
ARSAGA $N_{frozen}=10$	0.293	1.404	6.806	2.615
ARSAGA $N_{frozen}=5$	0.293	1.407	6.785	2.616

$$\phi_3 = d - d_{\min} \geq 0 \tag{27}$$

$$\phi_4 = D_{\max} - D - d \geq 0 \tag{28}$$

$$\phi_5 = \frac{D}{d} - 3 \geq 0 \tag{29}$$

$$\phi_6 = \delta_{\mathrm{pm}} - \delta_{\mathrm{p}} \geq 0 \tag{30}$$

(3) Deflection constraint

$$\phi_7 = \frac{F_{\max} - F_{\mathrm{p}}}{K} - \delta_{\mathrm{w}} \geq 0 \tag{31}$$

(4) Buckling constraint

$$\phi_8 = \frac{F_{\mathrm{crit}}}{1.25} - F_{\max} \geq 0 \tag{32}$$

where τ is the maximum shear stress in the helical spring, l_{f} the free length, δ_{p} the preload deflection, K the spring constant, F_{crit} the critical buckling load, and ϕ_i the ith constraint. The other symbols are expressed in Table 4. The complete expressions for some terms in these constraints can be found in Siddall [20]. Two different cases are studied in this work. In case 1, the spring is guided such that buckling will not occur and the buckling constraint is not necessary. In case 2, the spring is unguided and the buckling constraint is included. The parameters used in this problem are the same as Table 1 except that the maximum generation number is changed to 100. For comparison, different GA-based methods are considered. For each different GA-based method, 100 independent simulations are executed. For case 1, the function values of 100 independent simulations by different GA-based methods are shown in Fig. 5, and the searched results are further summarized in Table 5. For comparison, the result obtained

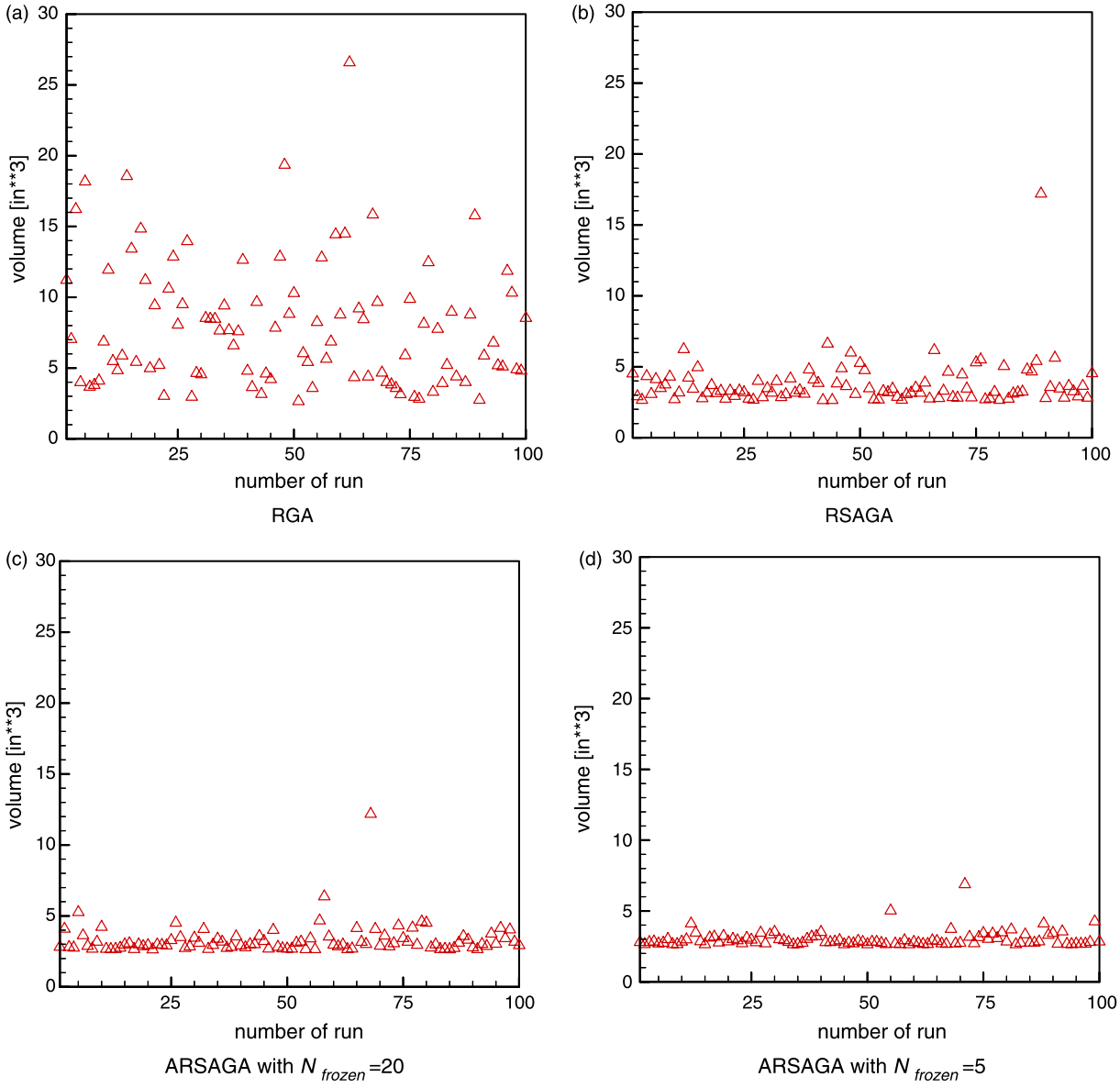

Fig. 6. Searched results with 100 independent simulations for case 2 helical spring.

Table 7
Comparison of the results by different methods for case 2 helical spring

Item	RGA	RSAGA	ARSAGA				
			50	30	20	10	5
Average	8.73	3.76	3.50	3.31	3.15	3.127	3.009
Standard deviation	7.156	1.650	1.257	1.106	1.060	0.818	0.562

from mathematical programming in [20] is included in Table 5 as accept value. In this table, the number of feasible solutions means the number of the searched solutions that do not violate any constraint in 100 independent simulations. The number of successful hits means the number of the searched solutions that are feasible and better than the accept value. It is evident that plain RGA has the worst performance. Although the searched solutions by RGA are feasible, they are not good enough, because there are only 35 successful hits in 100 independent simulations. Besides, its standard deviation is very high, and this indicates that plain RGA has poor reliability and stability in solution quality. Under the assistance of SA, the performances of RSAGA have been significantly improved. Since the standard deviation of the results is reduced from 3.458 to 0.153 and the average value is reduced from 4.21 to 2.68 that is less than the accept value, it can be said that compared to RGA, the searched solutions by RSAGA have high reliability and stability. Also, there are 96 successful hits in 100 independent simulations, and it implies that the solution quality has been improved significantly. In addition, if adaptive mechanisms are added, the performance of ARSAGA can be further improved. When a small stuck number is adopted, like 5, 10, and 20, the obtained results are excellent. A smaller stuck number indicates that the adaptive mechanisms are easier to be triggered. It is clearly evident that the solutions obtained by ARSAGA with small stuck numbers are extremely stable and reliable. The best values of design variables among 100 independent simulations by GA-based methods and the results obtained from [20] are listed in Table 6. For case 2, the function values of 100 independent simulations by different GA-based methods are shown in Fig. 6, and the searched results are further summarized in Table 7. Since case 2 was not studied in the original article, there is no reference value that can be used as the accept value, and the successful hit cannot be defined. However, from Fig. 6 and Table 7, one still could say that the performances of different GA-based methods are similar to case 1, and the proposed method is better than the other two GA-based methods, even though the buckling constraint is added. Comparing Tables 5 and 7, the helical spring under the buckling constraint needs more volume to resist the buckling stress. The best values of the design variables among 100 independent simulations by GA-based methods are also listed in Table 8. From these two cases, the proposed method with suitable stuck numbers has very excellent performance, and its solution is highly stable and reliable.

Table 8
Values of design variables by different methods for case 2 helical spring

Item	Wire diameter, d (in.)	Coil mean diameter, D (in.)	Number of coils, N	Volume of wire (in.^{3})
RGA	0.302	1.573	5.476	2.643
RSAGA	0.300	1.545	5.675	2.641
ARSAGA	0.300	1.539	5.710	2.636
$N_{\text{frozen}}=50$				
ARSAGA	0.300	1.536	5.725	2.631
$N_{\text{frozen}}=30$				
ARSAGA	0.300	1.533	5.745	2.630
$N_{\text{frozen}}=20$				
ARSAGA	0.300	1.541	5.967	2.632
$N_{\text{frozen}}=10$				
ARSAGA	0.301	1.561	5.542	2.636
$N_{\text{frozen}}=5$				

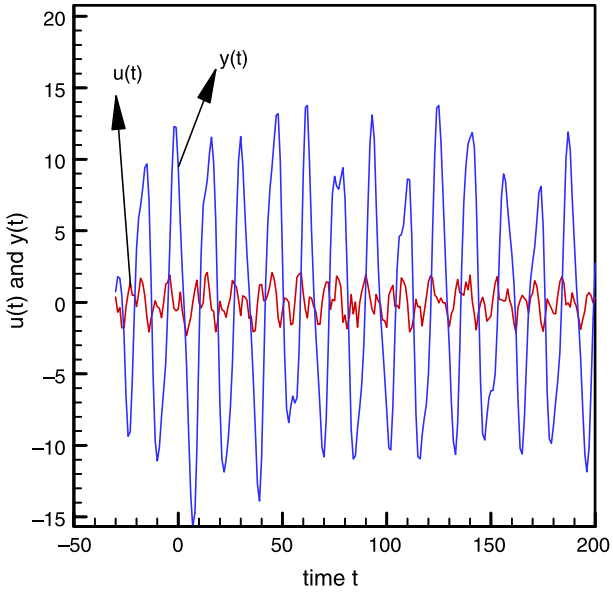

Fig. 7. Input function $u(t)$ and output function $y(t)$.

3.2. System identification

Consider a more complex and larger dynamic system that is described by an ARMAX model

$$A_{(q^{-1})}y_{(t)} = B_{(q^{-1})}u_{(t-d)} + C_{(q^{-1})}e_{(t)} \tag{33}$$

where

$$A_{(q^{-1})} = 1.0 - 1.5q^{-1} + 0.7q^{-2} \tag{34}$$

$$B_{(q^{-1})} = 1.0 + 0.5q^{-1} \tag{35}$$

$$C_{(q^{-1})} = 1.0 - 1.0q^{-1} + 0.2q^{-2} \tag{36}$$

The delay denoted as d is 1, and $u_{(t)}$, $y_{(t)}$, and q^{-1} are input function, output function, and backward shift operator, respectively. The $e_{(t)}$ is a normally distributed random noise with zero mean and unit variance. This system has been studied in Kristinsson [4]. Now, the goal is to determine the delay d and the parameters of the system polynomials, $A_{(q-1)}$ and $B_{(q-1)}$. The estimation of the simulated system can be written as

Table 9
Estimated parameter values for system identification example

Method	a_1	a_2	b_1	b_2	d	Population size
True value	−1.5	0.7	1.0	0.5	1	–
RGA average	−1.579024 (−5.27%)	0.771539 (10.22%)	0.858450 (−14.16%)	0.337939 (−32.4%)	1 (0%)	40
RGA best	−1.465600 (2.29%)	0.690902 (−1.30%)	0.723200 (−27.68%)	0.934659 (86.93)	1 (0%)	40
ARSAGA-2* average	−1.484747 (1.53%)	0.690962 (−1.29%)	0.931303 (−6.87%)	0.641542 (28.31%)	1 (0%)	40
ARSAGA-2* best	−1.501263 (−0.08%)	0.701868 (0.27%)	0.972266 (−2.77%)	0.524711 (4.94%)	1 (0%)	40
ARSAGA-1** average	−1.501607 (−0.11%)	0.701944 (0.28%)	0.984576 (−1.54%)	0.512971 (2.59%)	1 (0%)	40
ARSAGA-1** best	−1.498637 (0.091%)	0.698984 (−0.15%)	0.998635 (−0.14%)	0.507441 (1.49%)	1 (0%)	40
Ref. [21]: GA best	−1.506 (−0.4%)	0.7104 (1.49%)	0.9712 (−2.88%)	0.4939 (−1.22%)	1 (0%)	100
Ref. [12]: GASA best	−1.49 (0.6%)	0.69 (−1.43%)	1.0 (0%)	0.52 (4%)	5 (0%)	150
Ref. [12]: least mean-square technique	−1.31 (12.67%)	0.54 (22.86%)	1.1 (10%)	0.62 (24%)	5 (0%)	–

ARSAGA-2* denotes ARSAGA with $N_{\text{frozen}}=20$; ARSAGA-1** denotes ARSAGA with $N_{\text{frozen}}=10$.

$$\hat{A}_{(q^{-1})}\hat{y}_{(t)} = \hat{B}_{(q^{-1})}u_{(t-\hat{d})} \tag{37}$$

where

$$\hat{A}_{(q^{-1})} = 1.0 + a_1 q^{-1} + a_2 q^{-2} \tag{38}$$

$$\hat{B}_{(q^{-1})} = b_1 + b_2 q^{-1} \tag{39}$$

The estimated output function is denoted as $\hat{y}_{(t)}$, and the solution set to be searched is (a_1, a_2, b_1, b_2, d), where $a_1 \in [-2, 0]$, $a_2 \in [0, 2]$, $b_1 \in [0, 2]$, $b_2 \in [0, 2]$, and $d \in \{0, 1, 2, 3\}$. The difference between the output function and the estimated output function is denoted as

$$\eta_{(t)} = y_{(t)} - \hat{y}_{(t)} \tag{40}$$

and the objective function is defined as

$$F(t) = \sum_{i=0}^{m} [\eta_{(t-1)}]^2 \tag{41}$$

where m is the window size. The input function for the sample data is chosen as

$$u_{(t)} = \sin(t) - \sin(t/2.5) + \text{random}(-1, 1) \tag{42}$$

where random(−1,1) represents a random number chosen from the range from −1 to 1. Eq. (42) was used as the test input. The input and output function pairs are shown in Fig. 7. The parameter values are the same as those in Table 1. Note that the population size of 40 is rather small compared to the population size of 100 adopted in Jean and Chen [21] and the population size of 150 adopted in Tan et al. [12]. From the previous sections, the stuck number N_{frozen} is chosen to 10 or 20 for the proposed method, ARSAGA.

The final results of the estimated parameter values by RGA and ARSAGA are shown in Table 9, and the variation of the estimated parameter values with generation number is shown in Figs. 8–10. In the table and figures, the average values are obtained from five simulations with the same conditions. The best results among five independent simulations by different algorithms are also shown in the table. It is noted that instead of 1 the delay d is equal to 5 in [12]. For comparison, the results obtained by a variable-based genetic algorithm [21], genetic

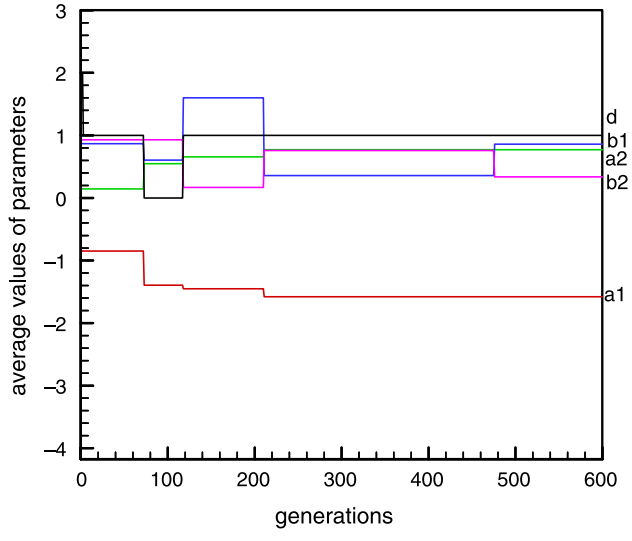

Fig. 8. Average parameter values of five simulations by RGA.

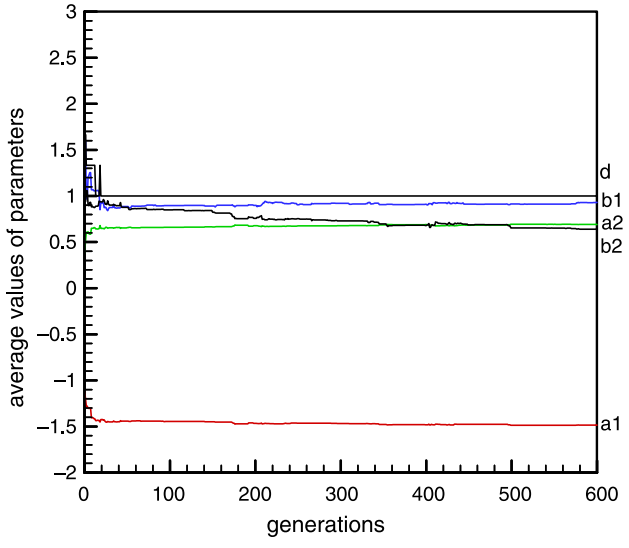

Fig. 9. Average parameter values of five simulations by ARSAGA with $N_{\text{frozen}}=20$.

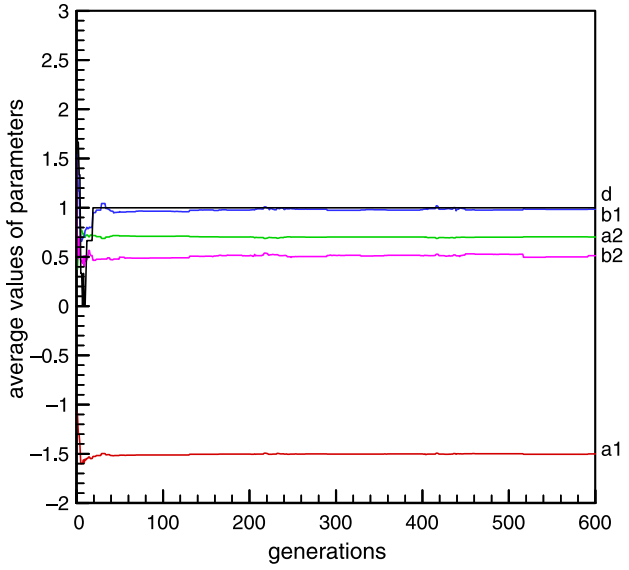

Fig. 10. Average parameter values of five simulations by ARSAGA with $N_{\text{frozen}}=10$.

algorithm with simulated annealing [12], and least mean-square technique [12] are also adopted. From Table 9, it shows that the results obtained by the GA-based techniques are always better than the results obtained by the least mean-square technique. Comparing these GA-based algorithms, the results obtained by plain RGA are not good. It even has a 32% error in b_2 by considering the average value. If ARSAGA with $N_{\text{frozen}}=20$ is used, the errors in the parameters are reduced. The estimation of a_1, a_2 and b_1 is very good, but the average error in b_2 is still very high. If ARSAGA with $N_{\text{frozen}}=10$ is applied, all estimations are in good agreement with the true values. The largest average error is only 2.59% that occurs in b_2. Comparing the best results by the variable-based genetic algorithm [21], the genetic algorithm with simulated annealing [12], and ARSAGA with $N_{\text{frozen}}=10$, the proposed algorithm with suitable stuck number is superior to the other two algorithms. Even though simulated annealing is included in the genetic algorithm in [12], its performance is not evidently better than that of the variable-based genetic algorithm. But, both the best solution and average solution by ARSAGA with stuck number $N_{\text{frozen}}=10$ are better than those by the other two methods. It can be said that ARSAGA with suitable stuck number has very high reliability and stability in searched solutions. Comparing Fig. 8 with Fig. 9, it is evident that ARSAGA has great improvement in hill-climbing ability. Comparing Fig. 9 with Fig. 10, it shows the performance of the adaptive mechanism. For a small N_{frozen}, the adaptive mechanism is more active and the convergence speed is faster. Therefore, for a complex and large system, the proposed algorithm, ARSAGA, with suitable stuck number is very powerful.

4. Conclusions

A novel hybrid algorithm, named ARSAGA, is proposed in this paper. It bases on a real-parameter genetic algorithm with a novel crossover operator and includes the merit of SA for improving the inherent shortages of GA, such as lack of hill-climbing capacity and slow convergence. To avoid premature convergence, adaptive mechanisms are also added in this novel algorithm. The proposed algorithm is different from other GA algorithms combined with SA because it simplifies the SA process and maintains the merits of each algorithm. Hence, it could not only save much computation time but also become more effective. Examples are chosen to check the performance of this hybrid algorithm. In the test functions, the excellent performances of the effective crossover mode, the mergence with SA, and the adaptive mechanism are verified. For the system identification example, that is a complex and large problem without constraint, the estimation of ARSAGA is in good agreement with the true values, if adaptive mechanisms are more active. As for the design of helical spring, it is a constrained optimization problem. Its result shows that the proposed method also provides solutions with high reliability and stability. Therefore, the proposed algorithm is significantly better than the other genetic algorithm-based methods or other methods discussed in this paper.

References

[1] Davis L. Handbook of genetic algorithm. New York: Van Nostrand Reinhold; 1991.
[2] Goldberg DE. Genetic algorithms in search, optimization, and machine learning. Reading: Addison-Wesley; 1989.
[3] Dumitrache I, Buiu C. Genetic learning of fuzzy controllers. Math Comput Simul 1999;49:13–26.
[4] Kristinsson K, Dumont GA. System identification and control using genetic algorithm. IEEE Trans Syst Man Cybern 1992;22(5): 1033–46.
[5] Nan J, Zhiye Z, Liqun R. Design of structural modular neural networks with genetic algorithm. Adv Eng Softw 2003;34:17–24.
[6] Van Laarhoven PJM, Aarts EHL. Simulats annealing: theory and applications.. Dordrecht, Holland: Reidel, Kluwer Academic Publishers; 1987.
[7] Jeong IK, Lee JJ. Adaptive simulated annealing genetic algorithm for system identification. Eng Appl Artif Intell 1996;9:523–32.
[8] Haupt RL, Haupt SE. Practical genetic algorithms. New York, NY: Wiley Interscience; 1998.
[9] Adler D. Genetic algorithm and simulated annealing: a marriage proposal. In: Proceeding of the IEEE international conference on neural network; 1993. p. 1104–9.
[10] Wong SV, Hamouda AMS. Optimization of fuzzy rules design using genetic algorithm. Adv Eng Softw 2000;31:251–62.
[11] Brown D, Huntley C, Spillane AA. A parallel genetic heuristics for the quadratic assignment problem. In: Proceeding of third international conference on neural networks; 1989. p. 406–15.
[12] Tan KC, Li Y, Murray-Smith DJ, Sharman KC. System identification and linearization using genetic algorithms with simulated annealing. In: Proceeding IEEE genetic algorithms in engineering system: innovations and applications, conference publication, vol. 414; 1995. p. 164–9.
[13] Adewuya AA. New methods in genetic search with real-valued chromosomes. Master's Thesis. Massachusetts Institute of Technology, Cambridge; 1996.
[14] Wright A. Genetic algorithm for real parameter optimization. In: Rawlins Gregory JE, editor. Foundations of genetic algorithm. San Mateo, CA: Morgan Kaufmann; 1991.

[15] Dejong KA. Analysis of the behavior of a class of genetic adaptive systems. PhD Thesis. University of Michigan, Ann Arbor, MI; 1975.
[16] Grefenstette JJ. Optimization of control parameters for genetic algorithms. IEEE Trans Syst Man Cybern 1986;SMC-16(1):122–8.
[17] Schaffer JD, Caruana RA, Eshelman LJ, Das R. A study of control parameters affecting online performance on genetic algorithms for function optimization. In: Proceeding of the third international conference of genetic algorithms; 1989. p. 51–60.
[18] Schwefel HP. Numerical optimization of computer models. New York, NY: Wiley; 1981.
[19] Huang YP, Huang CH. Real-valued genetic algorithms for fuzzy grey prediction system. Fuzzy Set Syst 1997;87:265–76.
[20] Siddall JN. Optimal engineering design-principles and applications. New York, NY: Marcel Dekker; 1982.
[21] Jean KT, Chen YY. A variable-based genetic algorithm. In: Proceedings of the IEEE international conference on systems, man, and cybernetics, vol. 2; 1994. p. 1597–601.