A Robust Stochastic Genetic Algorithm (StGA) for Global Numerical Optimization

Zhenguo Tu and Yong Lu

***Abstract*—Many real-life problems can be formulated as numerical optimization of certain objective functions. However, often an objective function possesses numerous local optima, which could trap an algorithm from moving toward the desired global solution. Evolutionary algorithms (EAs) have emerged to enable global optimization; however, at the present stage, EAs are basically limited to solving small-scale problems due to the constraint of computational efficiency. To improve the search efficiency, this paper presents a stochastic genetic algorithm (StGA). A novel stochastic coding strategy is employed so that the search space is dynamically divided into regions using a stochastic method and explored region-by-region. In each region, a number of children are produced through random sampling, and the best child is chosen to represent the region. The variance values are decreased if at least one of five generated children results in improved fitness, otherwise, the variance values are increased. Experiments on 20 test functions of diverse complexities show that the StGA is able to find the near-optimal solution in all cases. Compared with several other algorithms, StGA achieves not only an improved accuracy, but also a considerable reduction of the computational effort. On average, the computational cost required by StGA is about one order less than the other algorithms. The StGA is also shown to be able to solve large-scale problems.**

***Index Terms*—Evolutionary algorithms (EAs), global optimization, local selection, stochastic genetic algorithm (StGA), stochastic region.**

I. Introduction

THE NEED FOR numerical optimization algorithms arises from almost every field of engineering, science, and business. This is because an analytical optimal solution is difficult to obtain even for relatively simple application problems. A numerical algorithm is expected to perform the task of global optimization of an objective function. However, often, an objective function possesses numerous local optima, which could trap the numerical algorithms. The possibility of failing to locate the desired global solution increases with the increase of the problem dimension.

The recent developments in evolutionary algorithms (EAs), viz. genetic algorithm (GA), evolutionary strategy (ES), evolutionary programming (EP), and genetic programming (GP), have seen increasing applications in dealing with difficult global optimization problems [1]–[8]. EAs essentially are search algorithms based on the concepts of natural selection and survival of the fittest. They guide the evolution of a set of randomly selected individuals through a number of generations in approaching the global optimum solution. The distinctive advantages of EAs over other types of numerical methods include the following.

1) They only require information of the objective function itself, which can be either explicit or implicit. Other accessory properties such as differentiability or continuity are not necessary. As such, they are more flexible in dealing with a wide spectrum of problems.
2) Owing to the inherent implicit parallelism, EAs essentially work with building blocks, which increase exponentially as the evolution through generations proceeds. This results in an efficient exploitation of the given search space. Despite these superior features, EAs face the problem with high computational demand due to the generally slow evolutionary process.

So far, most of the successful applications of EAs are limited to problems with dimensions below 30 [9]–[12]. To promote broader applications of the algorithms and reduce the risk of getting a misguided solution, a more effective and efficient EA is needed.

In 1995, Krishnakumar *et al.* [4] proposed the use of a stochastic coding in binary GA (abbreviated as "StGA") with the intention to improve the efficiency. However, no evidence of significant improvement was reported. Since then, little has been done to further pursue the potential merits of this profound idea except a few applications (for example, Mulgund *et al.* applied StGA for air combat tactics optimization [7]), and no comparison has been reported to assess the actual performance of the algorithm. In fact, there still lacks a rigorous procedure for incorporating this novel algorithm. In view of the above, this paper is conducted to advance the idea of the StGA with the following two main objectives: 1) to develop a generic procedure for the implementation of the StGA and 2) to demonstrate the effectiveness and efficiency of the StGA as compared with other algorithms and to explore its ability in tackling large-dimension problems.

In this paper, the mechanism of the stochastic coding scheme is described first. The corresponding genetic operations, namely, selection, crossover, and mutation are discussed in detail. For the performance enhancement, a replacement strategy similar to that applied in the conventional GA is adopted. Necessary operational details and general guidelines on the actual coding are provided. The performance of the proposed StGA is assessed by carrying out optimization on 20 test functions of moderate dimensions (up to 30). Compared to some well-known global optimization algorithms such as ES, EP, and SA, StGA is shown to be able to achieve more accurate results and yet with a much reduced computational effort in almost all the cases examined.

Manuscript received July 18, 2003; revised December 17, 2003.

The authors are with the School of Civil and Environmental Engineering, Nanyang Technological University, 639798 Singapore (e-mail: cylu@ntu.edu.sg).

Digital Object Identifier 10.1109/TEVC.2004.831258

Finally, the capability of the StGA in solving large-dimension problems is examined through optimizing two functions of dimensions as high as 100. Comparing with the observations reported in the literature, StGA maintains a superior efficiency and effectiveness for such large-scale problems. It is, therefore, concluded that StGA is a promising technique in performing global optimization for practical applications.

II. Stochastic Genetic Algorithm (StGA)

The operation of the StGA stems from a totally different concept from the usual GA, particularly, in terms of the coding technique. The exclusive features of StGA include 1) each chromosome represents a stochastic region defined by a normal distribution; 2) these regions are dynamically adapted toward the most promising region; 3) no region in the search space is absolutely forsaken; and 4) the search region is not explicitly restrained.

In this paper, the following minimization problem with fixed boundaries is considered:

$$\min_{\vec{x}} f(\vec{x}) \quad \text{subject to} \quad \vec{B}_l \le \vec{x} \le \vec{B}_u$$

where $\vec{x} = (x_1, x_2, \cdots, x_m)$ is the variable vector in $\Re^M$, $f(\vec{x})$ denotes the objective function, and $\vec{B}_l = (B_{l1}, B_{l2}, \cdots, B_{lm})$, $\vec{B}_u = (B_{u1}, B_{u2}, \cdots, B_{um})$ represent, respectively, the lower and the upper bound of the variables (i.e., a predefined feasible solution space) such that the meaningful range of x_i is $[B_{li}, B_{ui}]$.

A. Stochastic Coding Mechanism

Most paradigms of GAs, regardless of what coding strategy is used, interpret each coded genotype "chromosome" (or "string") as one possible candidate solution. Hence, GAs actually explore the solution space for optimum in a point-by-point manner. This approach could be quite inefficient because in the early stage of evolution enormous effort may, thus, be wasted in evaluating the least significant digits of the gene that contribute little toward locating the most essential genes [13]. Although the later developed real-coding GA achieves a faster convergence rate, the computational efficiency remains to be a major constraint. To facilitate the solution of large-scale real-life problems, some special techniques have emerged, such as the dynamic coding GA [13], the messy floating-point GA [14], and the approximate function evaluation [15]. The applicability of these methods, unfortunately, is subjected to a variety of strict prerequisites on the physical problem itself.

The main purpose of the present the StGA is to achieve a highly efficient evolution with necessary robustness by incorporating an innovative stochastic coding method. The StGA codes each chromosome as a representative of a stochastic region described by a multivariate Gaussian distribution rather than a single candidate solution as in the conventional GA. In the StGA, a chromosome C_j comprises a binary string, M_j representing the mean vector of the Gaussian distribution, and a real number vector V_j representing the corresponding variable variances (see Fig. 1). The whole binary string is divided into

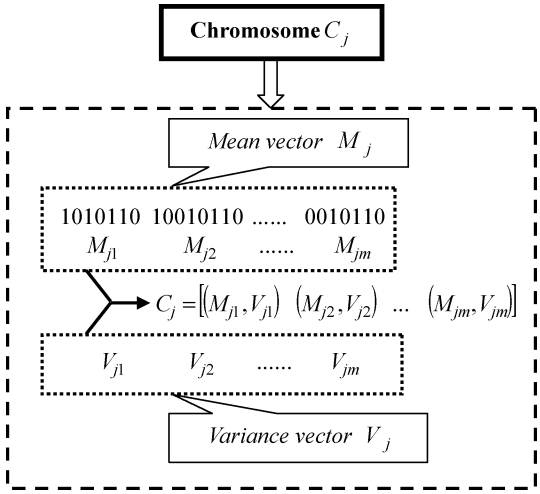

Fig. 1. Chromosome model for the StGA.

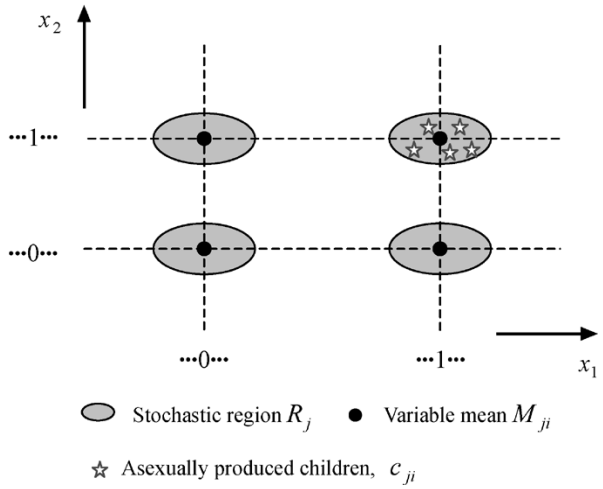

Fig. 2. Schematic illustration of stochastic regions in the StGA.

m substrings ($M_{ji}, i = 1, 2, \ldots, m$). Each substring indicates the geno-space coding of one particular variable x_i, and is associated with a variance V_{ji}. Thus

$$C_j = \lfloor (M_{j1}, V_{j1}) \quad (M_{j2}, V_{j2}) \quad \cdots \quad (M_{jm}, V_{jm}) \rfloor . \quad (1)$$

Fig. 2 illustrates schematically, the definition of stochastic regions decoded from C_j for a simple case involving two variables ($m = 2$). Coding the physical problem in this way facilitates the StGA to perform an efficient search by dynamically shifting emphasis to different favorable regions in the feasible space without abdicating any portion of the region. As such, the StGA effectively avoids the kind of failure as could be experienced in the normal dynamic coding scheme due to constantly throwing away potential regions regarded as "unpromising" in terms of their fitness.

As binary bits are used to code the variable mean values, a decision has to be made on the substring length for each variable. In conventional GA, the substring length (which represents the variable precision) is required to be long enough to describe the continuous space in order to achieve a desired accuracy. However, a longer substring length implies a substantial increase of the computational effort for GA, which can become

prohibitive as the problem dimension escalates. The StGA provides a possibility to reduce the computational demand by allowing a somewhat coarse division of the variable space without compromising the accuracy of the final solution. This is made possible by the fact that those points that are not covered by the binary string could also be approached by the StGA through numerical sampling within the stochastic regions. Despite this advantage, each substring still needs to have a sufficient number of bits to prevent blind genetic search. The choice of an appropriate number of bits will be discussed further in association with the actual coding for the numerical experiment in Section III.

B. Initialization of Population

Two initialization processes are required in the StGA, one for the mean vectors M_j, and the other for the variance vectors V_j. Generally speaking, M_j is initialized within the predefined searching space $[\overrightarrow{B_l}, \overrightarrow{B_u}]$ in a random manner. In case some *a priori* knowledge about the potential solution is available, M_j may be initialized so that its values are close to the guessed solution to put the algorithm at a good starting point. The initialization of the variance vectors V_j, however, is not so straightforward. The initialization of V_j bears great significance as it affects the explorative space in the StGA's evolution process. It is not possible though to derive a universal rule for the determination of adequate values for V_{ji}. Section III will provide some general guidelines on an empirical basis. Generally speaking, a larger space would require bigger V_{ji} to be efficient.

C. Selection Operation

The selection process is to pick up individuals from groups of parents and children in the preceding generation to form the mating pool for the next generation to evolve. Differing from other paradigms of GA, in StGA, an additional selection called "local selection" is required prior to the normal genetic selection.

1) Local Selection: Local selection essentially serves two purposes; one is to assess the fitness of the stochastic regions represented by each chromosome in the population, and the other is to implement the adaptation of the variance values V_{ji}. It operates as follows. First, within each region R_j (represented by chromosome C_j), N number of children ($c_{ji}, i = 1, \cdots, N$) are produced asexually through random sampling according to the predefined normal distribution (see Fig. 2). With a fitness evaluation, the best child c_j^* is chosen to actually represent this particular region, and the corresponding fitness value is regarded as that of chromosome C_j. The same c_j^* is then employed to redo the coding to supersede the parent mean vector M_j. Such a fitness evaluation scheme implies that in the StGA, the local selection of each generation is based on the fitness of individuals of its immediate foregoing generation. Meanwhile, the variance values V_{ji} are adapted following the 1/5 principle such that if at least one out of five asexually generated children (c_{ji}) results in improved fitness as compared with that of the mean vector M_j, the individual variance values are decreased; otherwise, they are increased. A more detailed description is given in Section III-B.

It should be noted that in the above process, some of the values in c_{ji} may violate the prescribed boundaries. Possible ways to tackle this problem include resampling until the breach of boundary constraints disappears; assigning such c_{ji} a very small fitness value; or replacing any violating value in c_{ji} with a random number drawn within its associated boundary. This paper applies a different strategy such that the out-of-boundary variable values are forced to equal their closest boundary values. This approach is deemed to be more appropriate for the StGA, because in the StGA, the individual stochastic regions (R_j) are small with respect to the whole search space; thus, the occurrence of boundary violation signifies that the StGA is exploring the boundary of the respective variables and so, it is rational to adopt the nearest boundary values to substitute those out-of-boundary values. Resampling, however, may be frustrated for large-dimension cases because of the prohibitively large number of trials. The method of allocating small fitness appears to be unreasonable as the chromosomes in question might also be good schemata.

2) Global Genetic Selection: Genetic selection is usually made within the parent population based on the fitness values of the individuals; an individual with better fitness value is more likely to survive into the mating pool. In the present stochastic GA, global selection is performed on the updated population resulting from the local selection. A number of selection methods exist, including the roulette wheel selection, the stochastic universal selection, and the ranking selection, among others [11]. The present study employs the so-called tournament selection, which is generally adopted in EP [16]. This selection method allows for a control of the selection pressure (the ratio of selection probability between the fittest and the least fit individuals). In general, the tournament selection produces one individual (chromosome) each time for the mating pool. It operates by first randomly picking T_n number of individuals from the parent population, then ranking them, and the best one is sent into the mating pool. The above procedure is repeated until the mating pool is full. It is noteworthy that T_n essentially plays the role of controlling the selection pressure in the sense that increasing T_n strengthens the selection pressure, and *vice versa*. Larger T_n promotes the GA convergence process but at the risk of leading to premature behavior; smaller T_n tends to reduce such a risk but at the cost of increased computational effort. Therefore, some kind of tradeoff decision is required.

D. Crossover Operation

As one of the major genetic operators, crossover is designed to produce offspring in the hope that better fitness is achieved through exchanging partial genetic information (the coding segment of the chromosome) of two parents. Before crossover, two parents should be prepared, say C_j and C_k (see Fig. 3). The detailed operation process differs slightly under different crossover schemes. The present study adopts the one-point crossover scheme. For the StGA, the variance terms V_{ji} should also undergo crossover because they are bonded together with the mean values M_{ji} to define the stochastic regions.

For the crossover, a cut site s is first randomly chosen within the length of M_j to determine which portion of binary bits should be exchanged. s could take place between two adjacent substrings M_{jp} and $M_{j(p+1)}$ (see s_2 in Fig. 3); in this case, we simply exchange all items (binary substrings and variance

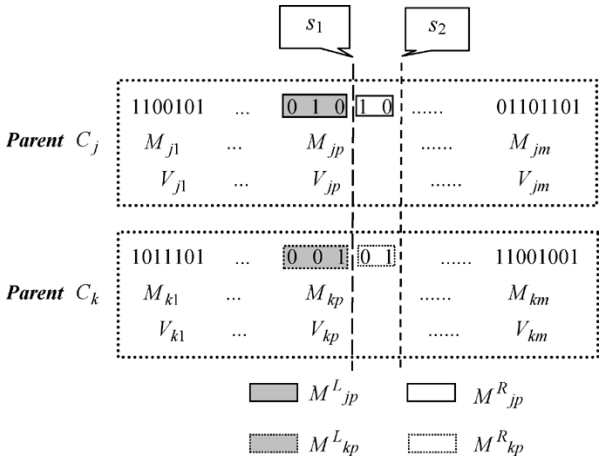

Fig. 3. Possible scenarios of crossover operation in the StGA (s_1: cutting through substrings; s_2: cutting between adjacent substrings).

values) at the right-hand side of s_2 between parent C_j and C_k to generate the two descendants $\overline{C}_j$ and $\overline{C}_k$

$$\begin{aligned}\overline{C}_j &= [(M_{j1}, V_{j1}) \quad \cdots \quad (M_{jp}, V_{jp}) \\ &\qquad \big(M_{k(p+1)}, V_{k(p+1)}\big) \quad \cdots \quad (M_{km}, V_{km})] \\ \overline{C}_k &= [(M_{k1}, V_{k1}) \quad \cdots \quad (M_{kp}, V_{kp}) \\ &\qquad \big(M_{j(p+1)}, V_{j(p+1)}\big) \quad \cdots \quad (M_{jm}, V_{jm})]. \end{aligned} \tag{2}$$

However, it is more likely that s falls within a particular substring, e.g., the pth substring M_{jp} and M_{kp}, as indicated by s_1 in Fig. 3. The crossover site, thus, splits M_{jp} and M_{kp} into four substrings M^L_{jp}, M^R_{jp}, M^L_{kp}, and M^R_{kp}. As such, the crossover on the corresponding variance terms V_{jp} and V_{kp} cannot be performed in a straightforward manner. For simplicity, this paper proposes a linear interpolation to produce two new offspring variance terms as

$$\begin{aligned} V^*_{jp} &= rV_{jp} + (1-r)V_{kp} \\ V^*_{kp} &= rV_{kp} + (1-r)V_{jp} \end{aligned} \tag{3}$$

where r is a random number between 0 and 1. By the above interpolation, it is more likely to retain the favorable stochastic regions discovered up to this step by the StGA. Thus, the two descendants $\overline{C}_j$ and $\overline{C}_k$ generated from crossover can be expressed as

$$\begin{aligned}\overline{C}_j &= \big[(M_{j1}, V_{j1}) \quad \cdots \quad \big(M^L_{jp} + M^R_{kp}, V^*_{jp}\big) \\ &\qquad \big(M_{k(p+1)}, V_{k(p+1)}\big) \quad \cdots \quad (M_{km}, V_{km})\big] \\ \overline{C}_k &= \big[(M_{k1}, V_{k1}) \quad \cdots \quad \big(M^L_{kp} + M^R_{jp}, V^*_{kp}\big) \\ &\qquad \big(M_{j(p+1)}, V_{j(p+1)}\big) \quad \cdots \quad (M_{jm}, V_{jm})\big]. \end{aligned} \tag{4}$$

In general, each parent chromosome in the population undergoes crossover only with certain probability p_c. For the binary coding, Dejong suggested taking the value of p_c between 0.7 and 0.9 [18]. Trial analyses in the experimental phase of this paper tends to support a use of $p_c = 0.85$.

E. Mutation Operation

Mutation is applied to increase the diversity of the population so as to enhance the chance for GA to escape from local optima. The actual process for mutation depends on the coding scheme. In binary coding, mutation only performs 1-bit flip, i.e., the bit value changes from 0 to 1 or from 1 to 0. Mutation happens on each individual binary bit with a probability of p_m, and this probability should be kept small; otherwise, the favorable building blocks (schemata) discovered so far by GA will be exhaustively destroyed, which implies failure. Based on Dejong's suggestion and the experience from trial analyses, p_m is taken in the range of [0.01, 0.025] for the various optimization cases described later in this paper. It should be pointed out that only the binary strings, which code the mean values of the stochastic regions, undergo mutation. The variance terms are not subject to mutation because they are adapted in connection with the properties of the stochastic regions.

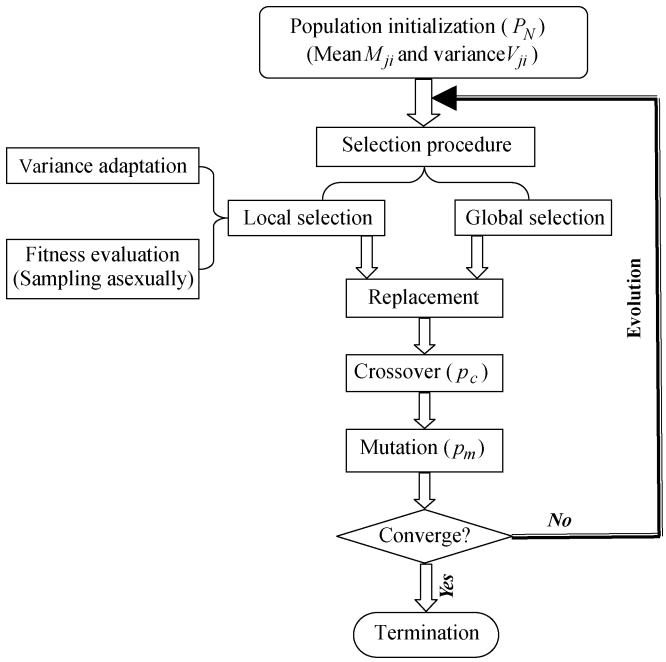

Fig. 4. General flowchart of the StGA execution.

F. Replacement Operation and Termination of Evolution

Replacement is a necessary operation to enhance the capacity of the StGA with particular regard to the possibility that an evolved offspring may be less fit than its parents and, consequently, degrade the algorithm's performance. It operates according to the so-called elitism strategy such that a small portion of the top ranking individuals from the parent generation is taken to substitute the same number of the least fit individuals in the offspring generation. As a result, the performance curve (fitness versus generation) of the StGA becomes monotonously increasing. The termination of evolution may be decided by certain criteria, e.g., a preset fitness value for the best individual or a prescribed maximum number of generations, depending on the nature of the underlying problems.

The general sequence of the StGA is summarized by a flowchart in Fig. 4. Further details on the actual operations will be given in Section III.

TABLE I
LIST OF 20 TEST FUNCTIONS (n = PROBLEM DIMENSION, $f_{\min}$ = MINIMUM FUNCTION VALUE, SD = PRESCRIBED SEARCH DOMAIN)

Test functions	n	SD	f_{min}
$f_1(x)=\sum_{i=1}^{n} x_i^{\,2}$	30	$[-100,100]^n$	0
$f_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	$[-10,10]^n$	0
$f_3(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i} x_j)^2$	30	$[-100,100]^n$	0
$f_4(x)=\max_i(\lvert x_i\rvert,1\le i\le n)$	30	$[-100,100]^n$	0
$f_5(x)=\sum_{i=1}^{n-1}\left[100(x_{i+1}-x_i^{\,2})^2+(x_i-1)^2\right]$	30	$[-30,30]^n$	0
$f_6(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	30	$[-100,100]^n$	0
$f_7(x)=\sum_{i=1}^{n}-ix_i^{\,4}+random[0,1)$	30	$[-1.28,1.28]^n$	0
$f_8(x)=\sum_{i=1}^{n}-x_i\sin(\sqrt{x_i})$	30	$[-500,500]^n$	-12569.5
$f_9(x)=\sum_{i=1}^{n}\left[x_i^{\,2}-10\cos(2\pi x_i)+10\right]$	30	$[-5.12,5.12]^n$	0
$f_{10}(x)=-20\exp(-0.2\sqrt{1/30\sum_{i=1}^{n}x_i^2})-\exp(1/30\sum_{i=1}^{n}\cos 2\pi x_i)$	30	$[-32,32]^n$	0
$f_{11}(x)=1/4000\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(x_i/\sqrt{i})+1$	30	$[-600,600]^n$	0
$f_{12}(x)=\pi/30\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$ *	30	$[-50,50]^n$	0
$f_{13}(x)=1/10\{10\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+(x_n-1)^2[1+\sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$ *	30	$[-50,50]^n$	0
$f_{14}(x)=\left[1/500+\sum_{j=1}^{25}(j+\sum_{i=1}^{2}(x_i-a_{ij})^6)^{-1}\right]^{-1}$ *	2	$[-65.536,65.536]^n$	1
$f_{15}(x)=\sum_{i=1}^{11}\left[a_i-x_1(b_i^2+b_ix_2)/b_i^2+b_ix_3+x_4\right]^2$ *	4	$[-5,5]^n$	$3.075e^{-4}$
$f_{16}(x)=4x_1^2-2.1x_1^4+1/3x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	$[-5,5]^n$	-1.0316285
$f_{17}(x)=(x_2-5.1/4\pi^2x_1^2+5/\pi x_1-6)^2+10(1-1/8\pi)\cos(x_1)+10$	2	$[-5,10]\times[0,15]$	0.398
$f_{18}(x)=-\sum_{i=1}^{5}\left[(x-a_i)(x-a_i)^T+c_i\right]^{-1}$ *	4	$[0,10]^n$	-10.1422
$f_{19}(x)=-\sum_{i=1}^{7}\left[(x-a_i)(x-a_i)^T+c_i\right]^{-1}$ *	4	$[0,10]^n$	-10.3909
$f_{20}(x)=-\sum_{i=1}^{10}\left[(x-a_i)(x-a_i)^T+c_i\right]^{-1}$ *	4	$[0,10]^n$	-10.5300

* Detailed description of these functions is given in the appendix.

III. IMPLEMENTATION OF THE StGA AND NUMERICAL EXPERIMENTS

A. Test Functions

Numerical experiments are conducted to test the effectiveness and efficiency of the StGA. Twenty test functions from three categories [17] are selected, covering a broader range than in some other relevant studies for the purpose to demonstrate the robustness and reliability of the present algorithm.

Table I lists the 20 test functions and their key properties. These functions can be divided into three categories of different complexities. f_1–f_7 are unimodal functions, which are relatively easy to optimize, but the difficulty increases as the problem dimension goes high. f_8–f_{13} are multimodal functions

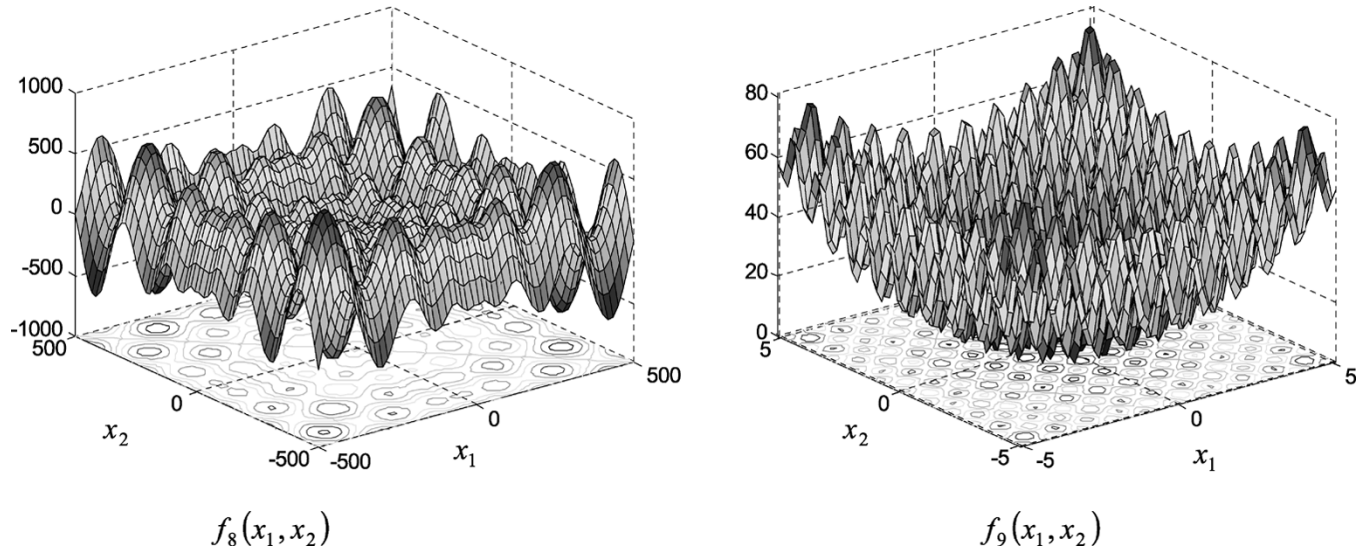

Fig. 5. Graphs of f_8 and f_9 with a dimension of 2.

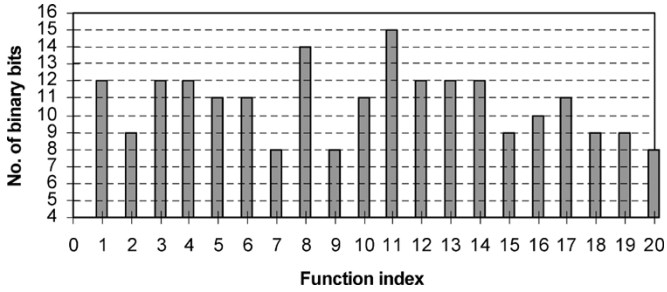

Fig. 6. Number of binary bits used in coding the variables of the test functions.

with many local optima, and they represent the most difficult class of problems for many optimization algorithms. As an example, Fig. 5 shows the surface landscapes of f_8 and f_9 when the dimension is set to 2. f_{14}–f_{20} are likewise multimodal functions, but they only contain a few local optima. It is interesting to note that some functions possess rather unique features. For instance, f_6 is a discontinuous step function having a single optimum; f_7 is a noisy quartic function involving a uniformly distributed random variable within [0, 1].

Generally speaking, for unimodal functions the convergence rates are of main interest as optimizing such functions to a satisfactory accuracy is not a major issue. For multimodal functions, however, the quality of the final results is more crucial since it reflects the StGA's ability in escaping from local deceptive optima and locating the desired near-global solution.

B. Numerical Implementation of the StGA

When implementing the StGA, a proper setup of the key parameters is required. First of all, the variable resolutions used in the coding process should be kept reasonably high so that the nearby region around the actual global solution can be approached by the StGA. A judgment on this may be reached with the help of a trial-and-error procedure. The next decision is on two important StGA-specific parameters, namely, the initial variance values V_{ji} and the adaptation step values δ_i for each variance term. In this paper, V_{ji} is initialized through a uniform

TABLE II
StGA PARAMETER SETTINGS AND ESTIMATED COMPUTATIONAL EFFORT (NUMBER OF FUNCTION EVALUATIONS)

TF	NP	NS	NG	MNFE	TF	NP	NS	NG	MNFE
f_1	30	5	200	30,000	f_{11}	50	5	210	52,500
f_2	22	5	160	17,600	f_{12}	20	5	80	8,000
f_3	40	5	115	23,000	f_{13}	20	5	160	16,000
f_4	40	5	160	32,000	f_{14}	20	5	8	800
f_5	50	5	180	45,000	f_{15}	40	5	150	30,000
f_6	20	5	15	1,500	f_{16}	20	5	40	4,000
f_7	30	5	170	25,500	f_{17}	20	5	50	5,000
f_8	20	5	15	1,500	f_{18}	20	5	100	10,000
f_9	30	5	190	28,500	f_{19}	20	5	48	4,800
f_{10}	40	5	50	10,000	f_{20}	20	5	85	8,500

TF: Test function **NP**: Population size **NS**: Number of asexually produced children
NG: Number of generations **MNFE**: Mean number of function evaluations

random draw within a prescribed range, denoted by R_i^V, while δ_i is determined with reference to R_i^V. It is worth pointing out that the adaptation of V_{ji} is basically a linear process in that the adaptation step δ_i vary only within a preset small range in the evolution procedure. Theoretically speaking, too large a δ_i could degrade the StGA performance because there could be insufficient region exploitation, whereas too small a δ_i would also slow down the evolution process due to the fact that in early generations much effort is spent exploring some unpromising regions. In this respect, a series of preliminary experimental studies have been conducted for the purpose of providing a rough guideline for the choice of R_i^V and δ_i. In general, the following empirical formulae may be considered in setting R_i^V and δ_i:

$$R_i^V = \left(\frac{1}{120} \sim \frac{1}{80}\right)[B_{ui} - B_{li}] \quad \delta_i = \left(\frac{2}{100} \sim \frac{5}{100}\right) V_i^R$$

TABLE III
COMPARISON OF OPTIMIZATION RESULTS AND COMPUTATIONAL EFFORT BETWEEN THE StGA AND FEP

TF	Computational effort (MNFE)		Optimization results: Mean best μ_{NG} (Variance σ_{NG})	
	StGA	FEP	StGA	FEP
f_1	30,000	150,000	2.45×10^{-15} (5.25×10^{-16})	5.7×10^{-4} (1.3×10^{-4})
f_2	17,600	200,000	2.03×10^{-7} (2.95×10^{-8})	8.1×10^{-3} (7.7×10^{-4})
f_3	23,000	500,000	9.98×10^{-29} (6.9×10^{-29})	1.6×10^{-2} (1.4×10^{-2})
f_4	32,000	500,000	2.01×10^{-8} (3.42×10^{-9})	0.30 (0.50)
f_5	45,000	2×10^6	0.04435 (0)	5.06 (5.87)
f_6	1,500	150,000	0.0 (0.0)	0.0 (0.0)
f_7	25,500	300,000	8.4×10^{-4} (1.0×10^{-3})	7.6×10^{-3} (2.6×10^{-3})
f_8	1,500	900,000	-12569.5 (0.0)	-12554.5 (52.6)
f_9	28,500	500,000	4.42×10^{-13} (1.14×10^{-13})	4.6×10^{-2} (1.2×10^{-2})
f_{10}	10,000	150,000	3.52×10^{-8} (3.51×10^{-9})	1.8×10^{-2} (2.1×10^{-3})
f_{11}	52,500	2 00,000	2.44×10^{-17} (4.54×10^{-17})	1.6×10^{-2} (2.2×10^{-2})
f_{12}	8,000	150,000	8.03×10^{-7} (1.96×10^{-14})	9.2×10^{-6} (3.6×10^{-6})
f_{13}	16,000	150,000	1.13×10^{-5} (4.62×10^{-13})	1.6×10^{-4} (7.3×10^{-5})
f_{14}	800	10,000	1.0 (0.0)	1.22 (0.56)
f_{15}	30,000	400,000	3.1798×10^{-4} (4.7262×10^{-6})	5.0×10^{-4} (3.2×10^{-4})
f_{16}	4,000	10,000	-1.03034 (1.00×10^{-3})	-1.0300 (4.9×10^{-7})
f_{17}	5,000	10,000	0.3986 (6.00×10^{-4})	0.3980 (1.5×10^{-7})
f_{18}	10,000	10,000	-9.828 (0.287)	-5.52 (1.59)
f_{19}	4,800	10,000	-10.40 (0.0)	-5.52 (2.12)
f_{20}	8,500	10,000	-10.450 (0.037)	-6.57 (3.14)

where V_i^R takes a random value within R_i^V following a uniform distribution. It should be pointed out that the above proposal is not meant to be universally applicable and adjustment may be necessary in dealing with some particular problems.

To further demonstrate the implementation of the above procedure, the operation for the optimization of function f_8 is detailed in the following. Due to its unknown surface feature, each variable of the function is represented using as many as 14 binary bits in coding the variable mean, resulting in a division length of 0.10, which is deemed fine enough for the solution of this particular problem based on a preliminary analysis. The population size is set to 20 and, it is initialized with the variance terms being generated uniformly within R_i^V of [8.5, 12.0], while the adaptation steps of variances δ_i $(i = 1, 2\cdots, 30)$ take values from [0.20, 0.40] in a random manner. These settings are consistent with the aforementioned empirical formulae. Thus, a chromosome C_k in the initial population is shown in the equation at bottom of the page.

For the StGA to start, other pertinent parameters are set as follows: Tournament size in the global selection $= 2$; replacement rate $= 10\%$; probability for crossover $= 0.85$; and probability for mutation $= 0.02$.

It should also be mentioned that a proper choice of the variable division length (grid resolution) will depend on the

$$\begin{array}{cccc} 11\,100\,101\,101\,011(M_{k1}) & 00\,110\,011\,001\,010(M_{k2}) & \cdots\cdots\cdots\cdots & 11\,110\,011\,011\,111(M_{k30}) \\ 10.22(V_{k1}) & 11.32(V_{k2}) & \cdots\cdots & 9.15(V_{k30}) \end{array}$$

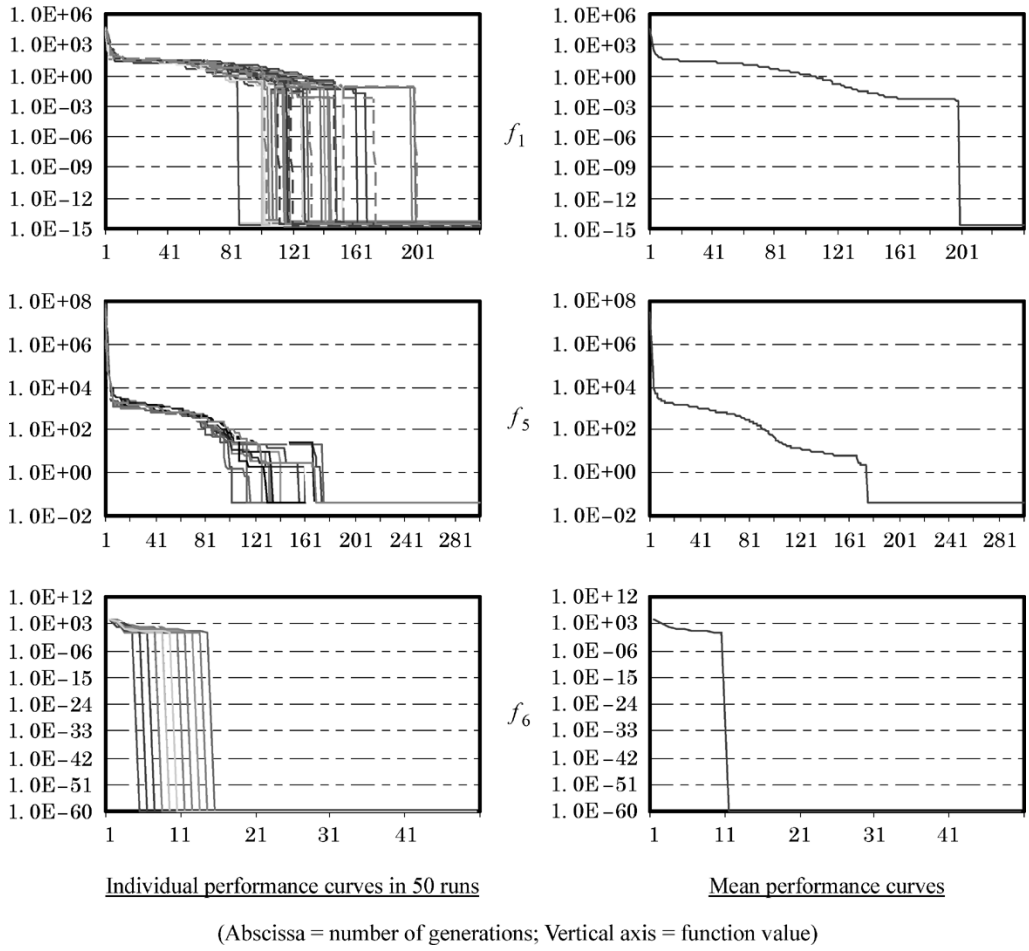

Fig. 7. Evolution curves of the StGA on functions f_1, f_5, and f_6.

landscape of the function (i.e., the sensitivity of the function with respect to each variable), as well as the size of the search domain, and the desirable resolution may vary among different variables. For simplicity, this paper adopts a uniform resolution for all variables involved in each test function. Fig. 6 shows the number of binary bits used in coding each variable in optimizing these test functions.

IV. Performance Assessment of the StGA and Comparison With Other Algorithms

The performance of the StGA is evaluated based on the optimization results on the 20 test functions as compared with some existing global optimization algorithms. For each test function, 50 runs with different seeds from the random number generator are performed to observe the consistency of the outcome. At each generation, the mean value of the best fit individuals from all 50 runs μ_i ($i = 1, \cdots, NG$, where NG is the maximum number of generation), is computed to plot the evolution curve, while the standard deviation of the best fit individuals from the last generation σ_{NG} is used to indicate the consistency of the algorithm. Generally speaking, a small σ_{NG} will signify a good consistency, while a large σ_{NG} may imply certain deficiency. The last mean value μ_{NG} represents the finally evolved optimum from the StGA and, hence, is used together with σ_{NG} to represent the optimization results in the comparison. Table II summarizes the key parameter settings of the StGA for each test function, including the population size, the number of asexually generated children in the local selection, and the number of generations. From these parameters, the number of function evaluations, which serves as a measure of the computational effort in this paper, is calculated and they are also shown in Table II.

A. Existing Algorithms for Comparison

There exist a number of global optimization algorithms suitable for continuous problems. For the present comparison, the following well-known algorithms are considered, and they will be applied for all or some of the 20 selected test functions depending on their specialized purposes.

1) Conventional Evolutionary Programming (CEP) with different mutation operators [19], namely, a) Gaussian mutation operator (CEP/GMO), designed for fast convergence on convex function optimization; b) Cauchy mutation operator (CEP/CMO), aimed for effective escape from the local optima; and c) mean mutation operator (CEP/MMO), which is a linear combination of Gaussian mutation and Cauchy mutation.
2) Fast Evolutionary Programming (FEP) [17]: FEP essentially employs a CMO but incorporates the GMO in an effective way.

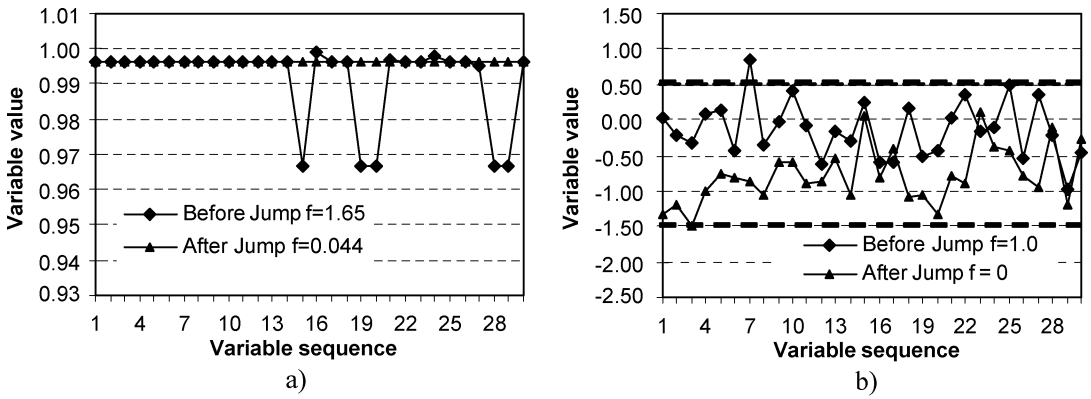

Fig. 8. Typical best variable sets evolved during the StGA optimization around a "jump" of function values. (a) Optimization of function f_5. (b) Optimization of function f_6.

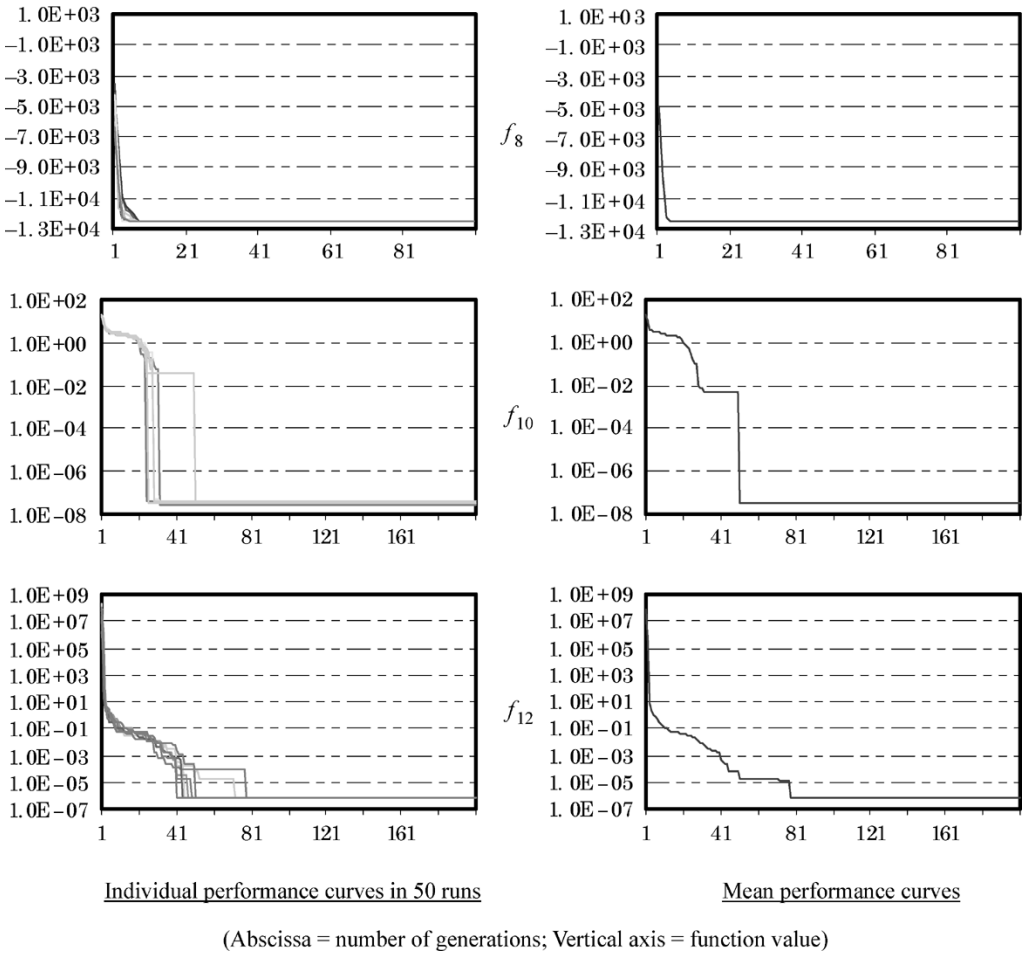

Fig. 9. Evolution curves of the StGA on functions f_8, f_{10}, and f_{12}.

3) Fast Evolution Strategy (FES) [10]: FES applies Cauchy mutation in the evolution strategies to generate each new generation.
4) Evolutionary Optimization (EO) [21]: EO uses a mutation operator and a selection scheme to evolve a population.
5) Particle Swarm Optimization (PSO) [21]: PSO is a new evolutionary computing scheme; it explores the insect swarm behavior.

Among these algorithms, of particular interest for the present comparison purpose is FEP proposed by Yao *et al.* [17]. The invention of the earlier CEP was dedicated to the global optimization of continuous problems, and it proved to be quite successful. FEP further enhances the capacity of CEP. Hence, a comparison with FEP will effectively demonstrate the global optimization capability of the present StGA.

B. Comparison Between the StGA and Other Algorithms

1) Comparison With FEP: Table III presents the optimization results obtained by the StGA in comparison with those from FEP. As can be seen, the StGA is able to locate the near-optimal

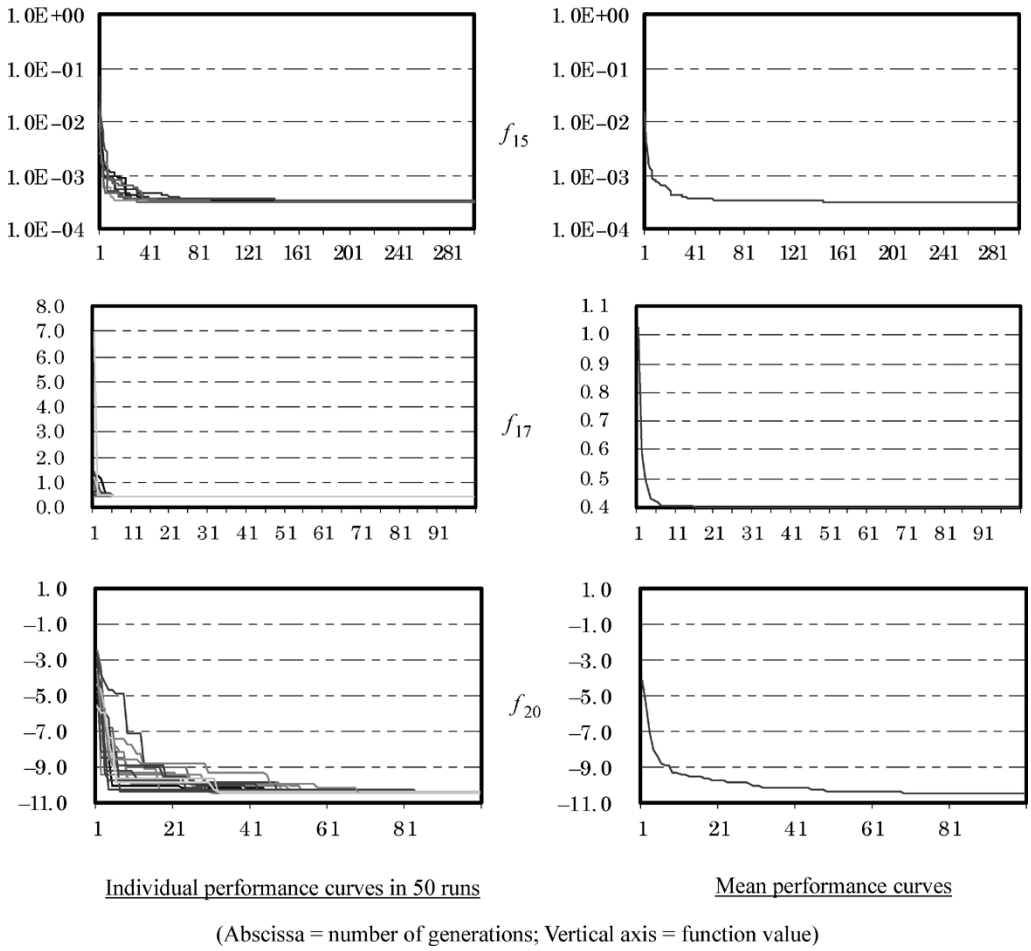

Fig. 10. Evolution curves of the StGA on functions f_{15}, f_{17}, and f_{20}.

solutions for all the 20 test functions with relatively small variance, indicating that the algorithm is both effective and statistically stable. Comparing with FEP, the StGA achieves generally much better optimization accuracy, while the required computational effort is reduced considerably.

For unimodal functions f_1–f_7, the StGA is able to obtain practically perfect optimization results, while FEP has difficulty with functions f_4 and f_5, and the accuracy for the remaining functions is also less good than the StGA. From the evolution curves (the best individual versus number of generations) shown in Fig. 7, it can be observed that the StGA can quickly converge toward the optima (zero value in these cases) within a relatively small number of generations. Taking the case of f_1 as an example, several orders of reduction of the function value, from 10^6 to 10^0, takes about 100 generations, whereas with FEP, 800 generations are required to achieve a similar result. It is interesting to note that for function f_6 the convergence is extremely fast and the correct optimum is found with only about 12 generations. In contrast, FEP requires 1500 generations to converge for this particular function. In this case, the StGA saves the computational effort by more than 100 times.

It is noted from Figs. 7 and 9 that there appears to be a jump to a low function value on the evolution curves of functions f_1, f_5, f_6, and f_{10}. In fact, there are two different scenarios here; one represents a true jump, which is due to the property of the function itself such as f_6, as will be discussed later; another is not exactly a "jump" but appears so because of the use of logarithm scale for the function value, while the actual optimum is zero. In such a case, when the evolution is getting closer to the optimum, such as for f_1 from $9.54E-3$ to $2.31e-15$, it appears as a big "jump" on the log scale plot.

From a more general perspective, certain degree of "jump" on a GA evolution curve is not uncommon when the replacement method is applied, due to the fact that the GA could sometimes experience a few generations without achieving a better solution. Such a situation can be diagnosed from the evolving variable values before and after the jump, as illustrated in Fig. 8(a) for function f_5. This function has the true optimum equal to zero. At a step prior to the "jump," most variables (25 out of the total 30) are already very close to their optimal values. After a couple of more generations, the StGA also locates the near-optimal values of these last few variables and the function value, thus, decreases from 1.65 to 0.044. Similar situations also happen to the optimization of functions f_1 and f_{10}. Function f_6 belongs to a different category in that it possesses a solution region instead of a single optimal point. All variables have the same optimal region of $[-1.5, 0.5]$, as shown in Fig. 8(b) between the two thick dashed lines. It can be seen that before the jump, all the 30 variables except the seventh variable are already in their optimal region; and when

TABLE IV
COMPARISON OF PERFORMANCE BETWEEN StGA AND FES

TF	MNFE		Mean best μ_{NG} (Variance σ_{NG})	
	StGA	FES	**StGA**	FES
f_8	1,500	900,030	−12569.5 (0.0)	−12556.4 (32.53)
f_9	28,500	500,030	4.42×10^{-13} (1.14×10^{-13})	0.16 (0.33)
f_{10}	10,000	150,030	3.52×10^{-8} (3.51×10^{-9})	1.2×10^{-2} (1.8×10^{-3})
f_{11}	52,500	200,030	2.44×10^{-17} (4.54×10^{-17})	3.7×10^{-2} (5.0×10^{-2})
f_{12}	8,000	150,030	8.03×10^{-7} (1.96×10^{-14})	2.8×10^{-6} (8.1×10^{-7})
f_{13}	16,000	150,030	1.13×10^{-5} (4.62×10^{-13})	4.7×10^{-5} (1.5×10^{-5})

TABLE V
PERFORMANCE COMPARISON AMONG StGA, PSO, AND EO

TF	MNFE			Mean best μ_{NG} (Variance σ_{NG})		
	StGA	PSO	EO	**StGA**	PSO	EO
f_1	30,000	250,000	250,000	2.45×10^{-15} (5.25×10^{-16})	11.175 (1.3208)	9.8808 (0.9444)
f_5	45,000	250,000	250,000	0.04435 (0)	1911.598 (374.2935)	1610.39 (293.5783)
f_9	28,500	250,000	250,000	4.42×10^{-13} (1.14×10^{-13})	47.1354 (1.8782)	46.4689 (2.4545)
f_{11}	52,500	250,000	250,000	2.44×10^{-17} (4.54×10^{-17})	0.4498 (0.0566)	0.4033 (0.0436)

the optimal region of this variable is also located, the function value jumps from 1 to 0.

For functions f_8–f_{13} which feature numerous local optima, the results shown in Table III clearly indicate that the StGA can identify the actual optima of these functions with good accuracy. Meanwhile, the efficiency as compared with FEP increases by 4 to 600 times in terms of the number of function evaluations. Fig. 9 shows typical evolution curves for functions f_8, f_{10}, and f_{12}, which demonstrate that the StGA behaves in a very stable manner over the 50 runs, despite the numerous local optima in these functions.

For the seven multimodal functions with fewer number of local optima (f_{14}–f_{20}), generally speaking, the StGA also exhibits a superior performance over FEP. Fig. 10 depicts the evolution curves for functions f_{15}, f_{17}, and f_{20}. Particularly noteworthy is the case of the function family f_{18}–f_{20}, which differ only in the number of terms in the summation, although FEP appears to be unable to approach the optima for these functions, the StGA maintains a satisfactory performance.

2) Comparison With CEP, FES, EO, and PSO: The performance of the StGA is further compared with some other well-established algorithms such as CEP, FES, ESA, and PSO. Since from the literature the optimization results using these algorithms are available only for some of the 20 test functions, the comparison will be made accordingly. The comparison results are summarized in Tables IV–VI.

From Table IV, it can be seen that FES generally can achieve satisfactory optimization results for the listed functions (expect f_9), but the StGA exhibits more accurate results, while the required computational effort is only about 1/10 of that required by FES. Results shown in Tables V and VI indicate that, while the StGA maintains a consistent and satisfactory performance,

TABLE VI
PERFORMANCE COMPARISON AMONG StGA, GMO, AND MMO

TF	MNFE				Mean best μ_{NG} (Variance σ_{NG})			
	StGA	CEP/GMO	CEP/CMO	CEP/MMO	**StGA**	CEP/GMO	CEP/ GMO	CEP/MMO
f_1	30,000	1500,00	1500,00	1500,00	2.45×10^{-15} (5.25×10^{-16})	3.09×10^{-7}	3.07×10^{-6}	9.81×10^{-7}
f_2	17,600	250,000	250,000	250,000	2.03×10^{-7} (2.95×10^{-8})	1.99×10^{-3}	5.87×10^{-3}	3.23×10^{-3}
f_3	23,000	250,000	250,000	250,000	9.98×10^{-29} (6.9×10^{-29})	17.60	5.78	11.80
f_4	32,000	250,000	250,000	250,000	2.01×10^{-8} (3.42×10^{-9})	5.18	0.66	1.88
f_5	45,000	250,000	250,000	250,000	0.04435 (0)	86.70	114.0	63.8
f_7	25,500	250,000	250,000	250,000	0.008412 (0.001023)	12.20	9.42	9.53
f_9	28,500	250,000	250,000	250,000	4.42×10^{-13} (1.14×10^{-13})	120.0	4.73	9.52
f_{10}	10,000	1500,00	1500,00	1500,00	3.52×10^{-8} (3.51×10^{-9})	9.10	1.3×10^{-3}	7.49×10^{-4}
f_{11}	52,500	250,000	250,000	250,000	2.44×10^{-17} (4.54×10^{-17})	2.52×10^{-7}	2.2×10^{-6}	6.99×10^{-7}

TABLE VII
StGA PARAMETER SETTING AND REQUIRED COMPUTATIONAL EFFORT FOR OPTIMIZING f_5^{100} AND $g(x)$

TF	NP	NS	NG	MNFE
f_5^{100}	30	5	240	36,000
$g(x)$	22	5	130	14,300

PSO, EO, and CEP appear to be unable to approach the optima for most of the listed functions, even after spending a considerable computational effort.

C. Performance of the StGA in Solving Large-Scale Optimization Problems

In the preceding sections, the superior performance of the StGA as compared with other algorithms has been demonstrated by optimizing both unimodal and complex multimodal functions up to moderate dimensions (up to 30). In order to examine the performance of the StGA in handling large-scale problems, in this section, the algorithm is used to perform optimization for two functions having a dimension as high as 100. The first function is an expanded version of f_5 with dimension increased from 30 to 100, denoted as f_5^{100}. The global minimum of this function remains to be zero. The other function takes the form

$$g(x) = \frac{1}{m}\sum_{i=1}^{m}\left(x_i^4 - 16x_i^2 + 5x_i\right) \text{ s.t. } -10 \le x_i \le 10.$$

For any positive integer m, the global minimum of $g(x)$ is -78.3323, and it occurs at point $X_{\min} = (-2.9035, -2.9035, \cdots, -2.9035)^T$.

The optimization of f_5^{100} has been reported in [20] using an efficient evolutionary programming (EEP) algorithm, while the optimization of $g(x)$ has been performed in [22] using the enhanced simulated annealing algorithm (ESA). These previous results are used to compare with the StGA.

The key parameters used in the StGA for optimizing the above two functions are listed in Table VII. Table VIII compares the optimization results from the StGA with those from ESA and EEP. Once again, the StGA exhibits a superior performance. For function $g(x)$, the optimal value identified by the StGA is within 0.1% of the true optimum, while the EEP error is about 2%. For f_5^{100}, the superiority of the StGA is more obvious. Besides the accuracy, the computational cost using the StGA is only 1/8–1/4 of that using ESA or EEP. Fig. 11 shows the evolution curves of the StGA in optimizing f_5^{100} and $g(x)$.

TABLE VIII
COMPARISON OF OPTIMIZATION RESULTS AND COMPUTATIONAL EFFORT AMONG StGA, EEP, AND ESA

TF	MNFE		Mean best μ_{NG} (Variance σ_{NG})	
	StGA	ESA [1]/EEP[2]	StGA	ESA [1]/EEP[2]
f_5^{100}	36,000	150,000[1]	1.01 (1.1959)	17.10 [1] (NA)
$g(x)$	14,300	122,000[2]	-78.29368 (0.03)	-76.782 [2] (NA)

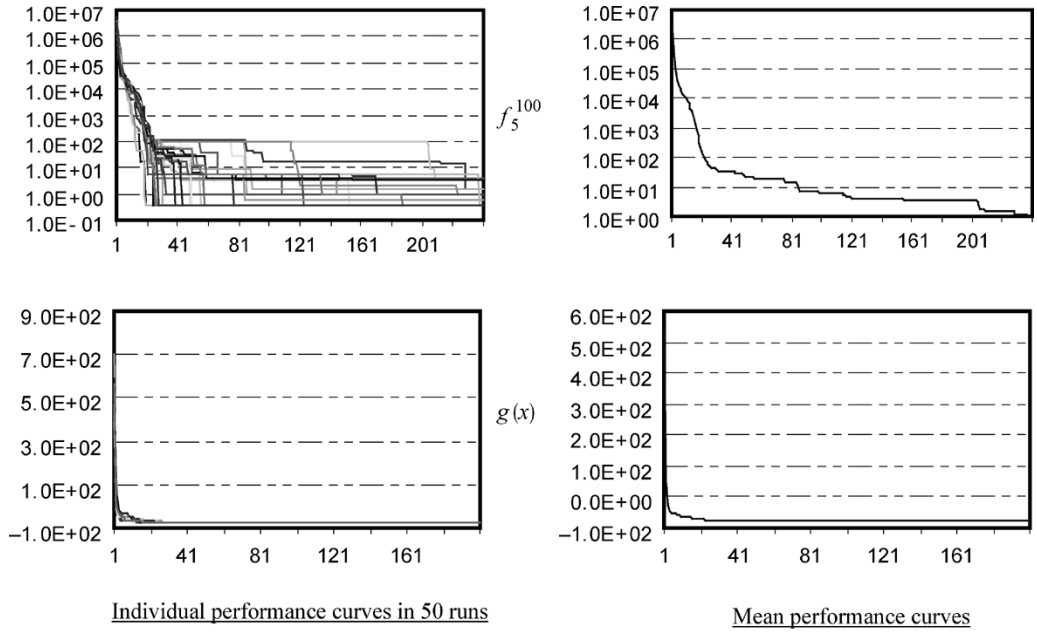

Fig. 11. Evolution curves of the StGA on functions f_5^{100} and $g(x)$.

V. CONCLUSION

In this paper, a stochastic genetic algorithm (StGA) is presented to deal with global optimization problems with continuous variables. The methodology involves a novel coding mechanism, in which the search space is dynamically divided into stochastic regions represented by a mean vector (coded in binary strings) and a variance vector. An effective crossover scheme is proposed such that when the cut site of crossover happens to fall within a substring representing a mean phenotype variable value, an interpolated variance is produced for crossover of the variance term. To further enhance the StGA performance, a similar replacement scheme as in the conventional genetic algorithms is incorporated into the operation of the StGA.

The algorithm is tested on 20 functions of moderate dimensions from three different categories. Results obtained from 50 trials for each function show that the StGA is able to find the near-global solution for all these test functions; moreover, the behavior of the algorithm is stable as indicated by a small variance among the 50 trial runs. Comparison of the StGA outcome with those from several other global optimization algorithms demonstrates that the StGA outperforms the other techniques with a dramatic improvement in terms of effectiveness, as well as efficiency. In general, the accuracy of the StGA increases by several orders of magnitude. For those functions, where other algorithms experience difficulties in approaching the optima, the StGA still exhibits a satisfactory performance. On average, the number of function evaluations required by the StGA is about one order less than the other algorithms.

The StGA is also tested to be capable of solving large dimension problems with a good efficiency.

TABLE IX
COEFFICIENTS APPEARING IN FUNCTION f_{15}

i	1	2	3	4	5	6	7	8	9	10	11
a_i	0.1957	0.1947	0.1735	0.1600	0.0844	0.0627	0.0456	0.0342	0.0323	0.0235	0.0246
b_i^{-1}	0.25	0.5	1	2	4	6	8	10	12	14	16

APPENDIX
DETAILED DESCRIPTION OF SOME TEST FUNCTIONS

A. f_{12} and f_{13} (Generalized Penalized Functions)

$$f_{12}(x)=\frac{\pi}{30}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_n-1)^2\right\}+\sum_{i=1}^{n}u(x_i,10,100,4)\quad -50\le x_i\le 50$$
$$\min(f_{12})=f_{12}(-1,-1,\cdots,-1)$$
$$f_{13}(x)=\frac{1}{10}\left\{10\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right]+(x_n-1)^2\left[1+\sin^2(2\pi x_n)\right]\right\}+\sum_{i=1}^{n}u(x_i,5,100,4)\quad -50\le x_i\le 50$$
$$\min(f_{13})=f_{13}(-1,-1,\cdots,-1)$$

where

$$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a,\\ 0, & -a\le x_i\le a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$$
$$y_i=1+\frac{1}{4}(x_i+1).$$

B. f_{14} (Shekel's Foxholes Function)

$$f_{14}(x)=\left[\frac{1}{500.}+\sum_{j=1}^{25}\left(j+\sum_{i=1}^{2}(x_i-a_{ij})^6\right)^{-1}\right]^{-1}$$
$$-65.536\le x_i\le 65.536,$$
$$\min(f_{14})=f_{14}(-32,-32)\approx 1$$

where

$$\left(a_{ij}=\begin{bmatrix}-32 & -16 & 0 & 16 & 32 & -32 & \cdots & 0 & 16 & 32\\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32\end{bmatrix}\right).$$

TABLE X
COEFFICIENTS IN FUNCTIONS f_{18}–f_{20}

i	a_{ij}, $j=1,2,\cdots,4$				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

C. f_{15} (Kowalik's Function) (Table IX)

$$f_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1\left(b_i^2+b_ix_2\right)}{b_i^2}+b_ix_3+x_4\right]^2$$
$$-5\le x_i\le 5$$
$$\min(f_{15})=f_{15}(0.1928,0.1908,0.1231,0.1358)=0.000\,307\,5.$$

D. f_{18}–f_{20} (Shekel's Family) (Table X)

$$f(x)=-\sum_{i=1}^{m}\left[(\mathbf{x}-\mathbf{a_i})(\mathbf{x}-\mathbf{a_i})^T+c_i\right]^{-1}$$

where

$$\mathbf{x}=(x_1,\quad x_2,\quad x_3,\quad x_4)^T\quad 0\le x_j\le 10\quad j=1,2\cdots 4$$

$f(x)$ with m equal to 5, 7, and 10, respectively, become f_{18}, f_{19}, and f_{20}. These three functions have five, seven, and ten local minima, respectively.

REFERENCES

[1] R. Tuccillo and A. Senatore, "A genetic algorithm-based approach to radial flow impeller design," *Comput. Fluid Dyn. Aeropropulsion ASME*, vol. 49, pp. 51–62, 1995.
[2] H. Hao and Y. Xia, "Vibration-based damage detection of structures by genetic algorithm," *J. Comput. Civil Eng.*, vol. 16, no. 3, pp. 222–229, 2002.

[3] M. I. Friswell, J. E. T. Penny, and S. D. Garvey, "A combined genetic and eigensensitivity algorithm for the location of damage in structures," *Comput. Structures*, vol. 69, pp. 547–556, 1998.
[4] K. Krishnakumar, R. Swaminathan, S. Garg, and S. Narayanaswamy, "Solving large parameter optimization problems using genetic algorithms," in *Proc. Guidance, Navigation, Contr. Conf.*, Baltimore, MD, 1995, pp. 449–460.
[5] S. R. Sathyanarayan, H. K. Birru, and K. Chellapilla, "Evolving nonlinear time-series models using evolutionary programming," in *Proc. Congr. Evolutionary Computation*, vol. 1, 1999, pp. 236–243.
[6] G. W. Greenwood, C. Lang, and S. Hurley, "Scheduling tasks in real-time systems using evolutionary strategies," in *Proc. 3rd Workshop Parallel Distributed Real-Time Systems*, 1995, pp. 195–196.
[7] S. Mulgund, K. Harper, K. Krishnakumar, and G. Zacharias, "Air combat tactics optimization using stochastic genetic algorithms," in *Proc. IEEE Int. Conf. Systems, Man, Cybern.*, vol. 4, 1998, pp. 3136–3141.
[8] H. Seywald, R. R. Kumar, and S. M. Deshpande, "Genetic algorithm approach for optimal control problems with linearly appearing controls," *J. Guidance, Control, Dynamics*, vol. 18, no. 1, pp. 177–182, 1995.
[9] W. W. Hager, W. Hearn, and P. M. Pardalos, Eds., *Large Scale Optimization: State of the Art*. Norwell, MA: Kluwer, 1994.
[10] X. Yao, Y. Liu, and G. M. Lin, "Fast evolutionary strategy," in *Proc. Evolutionary Programming VI*, P. J. Angeline, R. Reynolds, J. McDonnell, and R. Eberhart, Eds., 1997, pp. 151–161.
[11] D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[12] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. Berlin, Germany: Springer-Verlag, 1996.
[13] N. S. Nicol and K. B. Richard, "Dynamic Parameter Encoding for Genetic Algorithms," Univ. California at San Diego, La Jolla, CA, UCSD Tech. Rep. CS 90-175, 1992.
[14] K. Deb, "Binary and floating-point function optimization using messy genetic algorithms," Ph.D. Dissertation, Dept. Eng. Mech., Univ. Alabama, Tuscaloosa, AL, 1991.
[15] J. Grefenstette and M. Fritzpatrick, *Genetic Search With Approximate Function Evaluations*. Nashville, TN: Dept. Comput. Sci., Vanderbilt Univ., 1990.
[16] D. B. Fogel, "Evolving artificial intelligence," Ph.D. dissertation, Univ. California at San Diego, La Jolla, CA, 1992.
[17] X. Yao, Y. Liu, and G. M. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 82–102, July 1999.
[18] K. A. Dejong, "Analysis of the behavior of a class of genetic adaptive systems," Ph.D. Dissertation, Univ. Michigan, Ann Arbor, MI, 1975.
[19] K. Chellapilla, "Combining mutation operators in evolutionary programming," *IEEE Trans. Evol. Comput.*, vol. 2, pp. 91–96, Sept. 1998.
[20] J. H. Zhang and X. H. Xu, "An efficient evolutionary programming algorithm," *Comput. Oper. Res.*, vol. 26, pp. 645–663, 1999.
[21] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences," in *Proc. Evolutionary Programming VII*, V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds., 1998, pp. 601–610.
[22] P. Siarry, G. Berthiau, F. Durbin, and J. Haussy, "Enhanced simulated annealing for globally minimizing functions of many-continuous variables," *ACM Trans. Math. Software*, vol. 23, pp. 209–228, June 1997.

Zhenguo Tu received the B.Sc. and M.Eng. degrees from Shanghai Jiaotong University, Shanghai, China, in 1999 and 2001, respectively. He is currently working toward the Ph.D. degree with the School of Civil and Environmental Engineering, Nanyang Technological University, Singapore.

His current research interests include artificial intelligence algorithms, finite-element model updating, and structural health monitoring.

Yong Lu received the B.Eng. degree from Tongji University, Shanghai, China, the M.Eng. degree from Southeast University, Nanjing, China, and the Ph.D. degree from the National Technical University of Athens, Athens, Greece.

He is currently an Associate Professor with the School of Civil and Environmental Engineering, Nanyang Technological University, Singapore. His main research interests include structural identification and health monitoring using vibration data, smart structures and materials, and application of artificial intelligence algorithms in finite-element model updating, among others.