Hybrid Simplex-improved Genetic Algorithm for Global Numerical Optimization

REN Zi-Wu[1] SAN Ye[1] CHEN Jun-Feng[2]

Abstract In this paper, a hybrid simplex-improved genetic algorithm (HSIGA) which combines simplex method (SM) and genetic algorithm (GA) is proposed to solve global numerical optimization problems. In this hybrid algorithm some improved genetic mechanisms, for example, non-linear ranking selection, competition and selection among several crossover offspring, adaptive change of mutation scaling and stage evolution, are adopted; and new population is produced through three approaches, i.e. elitist strategy, modified simplex strategy and improved genetic algorithm (IGA) strategy. Numerical experiments are included to demonstrate effectiveness of the proposed algorithm.

Key words Genetic algorithm, simplex method, competition and selection,mutation scaling

1 Introduction

Genetic algorithm (GA) is a stochastic and parallel search technique based on the mechanics of natural selection, genetics and evolution, which was first developed by Holland in 1970s[1]. In recent years, GA has been widely applied to different areas such as fuzzy systems, neural networks, *etc.*[2] Although GA has become one of the popular methods to address some global optimization problems, the major problem of GA is that it may be trapped in the local optima of the objective function when the problem dimension is high and there are numerous local optima. This degradation in efficiency is apparent especially in applications where the parameters being optimized are highly correlated[3]. In order to overcome these flaws and improve the GA′s optimization efficiency, recent research works have been generally focused on two aspects. One is improvements upon the mechanism of the algorithm, such as modification of genetic operators, or the use of niche technique[4], *etc*; the other is combination of GA with other optimization methods, such as BFGS methods[5], simulated annealing (SA), *etc.*

In this paper, a hybrid simplex-improved genetic algorithm (HSIGA) is proposed to solve global numerical optimization problems. In this hybrid algorithm some improved genetic mechanisms are adopted, such as non-linear ranking selection, competition and selection among several crossover offspring, adaptive change of mutation scaling and adaptive stage evolution mechanism, to form an improved genetic algorithm (IGA). For further performance enhancement, the IGA algorithm is combined with the simplex method (SM) and the new population is generated through three approaches, *i.e.* elitist strategy, simplex strategy and IGA strategy. We investigate the effectiveness of this proposed algorithm by solving 10 test functions with high dimensions.

Received June 17, 2005; in revised form April 26, 2006
Supported by National Natural Science Foundation of P. R. China (60474069)

1. Control & Simulation centre, Harbin Institute of Technology, Harbin 150080, P.R.China 2. College of Computer & Information Engineering, Hohai University, Changzhou 213022, P.R. China

DOI: 10.1360/aas-007-0091

2 Hybrid simplex-improved genetic algorithm (HSIGA) for numerical optimization

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

$$\begin{aligned} &\text{minimize} && f(\boldsymbol{x}) = f(x_1, x_2, \cdots, x_n) \\ &\text{subject to} && x_i^{\min} \leqslant x_i \leqslant x_i^{\max} \quad (i = 1, 2, \cdots, n) \end{aligned} \tag{1}$$

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ is a variable vector in $\boldsymbol{R}^n$, $f(\boldsymbol{x})$ denotes the objective function, and $x_i^{\min}$ and $x_i^{\max}$ represent the lower and the upper bounds of x_i such that the meaningful range of x_i is $[x_i^{\min}, x_i^{\max}]$.

2.1 Improved genetic algorithm (IGA)

For the minimization problem like (1), we adopt real-code GA and firstly introduce IGA.

1) **Non-linear ranking selection operator**

In order to select some excellent chromosomes from the parent generation, non-linear ranking selection is adopted in this paper, which maps chromosome′s serial number in the queue to an expected selection probability.

With the population $pop = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_i, \cdots, \boldsymbol{x}_P\}$ of P chromosomes, we distribute the probability to each chromosome from the best to the worst by a non-linear function, so the selection probability of chromosome $\boldsymbol{x}_i$ is

$$\begin{cases} p(\boldsymbol{x}_i) = q'(1-q)^{i-1} \\ q' = \dfrac{q}{1-(1-q)^P} \end{cases} \tag{2}$$

where q is the selection probability of the best chromosome, i is the serial number of the chromosome.

After the selection probability of each chromosome is determined, the roulette wheel selection is adopted to select the excellent chromosome. Ranking selection need neither use the individual′s fitness nor transform the fitness scaling, which can prevent the premature convergence or stagnation phenomenon to a certain extent.

2) **Competition and selection among several crossover offspring**

In the natural evolution, parents often reproduce more than two offspring after crossover operation, and competition phenomenon between offspring of the same parents also always occurs. Illumined by this idea, competition and selection of the excellent among the several offspring is employed in the crossover operator. Different from the crossover operation of the simple genetic algorithm (SGA), four chromosomes are created firstly from the parents $\boldsymbol{x}_s = [x_1^s, x_2^s, \cdots, x_n^s]$ and $\boldsymbol{x}_t = [x_1^t, x_2^t, \cdots, x_n^t]$

$(s \neq t)$ according to the following formulae[2].

$$\boldsymbol{y}_1 = [y_1^1, y_2^1, \cdots, y_n^1] = \frac{\boldsymbol{x}_s + \boldsymbol{x}_t}{2} \tag{3}$$

$$\boldsymbol{y}_2 = [y_1^2, y_2^2, \cdots, y_n^2] = \boldsymbol{x}_{\max}(1-\omega) + \max(\boldsymbol{x}_s, \boldsymbol{x}_t)\omega \tag{4}$$

$$\boldsymbol{y}_3 = [y_1^3, y_2^3, \cdots, y_n^3] = \boldsymbol{x}_{\min}(1-\omega) + \min(\boldsymbol{x}_s, \boldsymbol{x}_t)\omega \tag{5}$$

$$\boldsymbol{y}_4 = [y_1^4, y_2^4, \cdots, y_n^4] = \frac{(\boldsymbol{x}_{\max} + \boldsymbol{x}_{\min})(1-\omega) + (\boldsymbol{x}_s + \boldsymbol{x}_t)\omega}{2} \tag{6}$$

$$\boldsymbol{x}_{\max} = [x_1^{\max}, x_2^{\max}, \cdots, x_n^{\max}] \tag{7}$$

$$\boldsymbol{x}_{\min} = [x_1^{\min}, x_2^{\min}, \cdots, x_n^{\min}] \tag{8}$$

where $\omega \in [0,1]$ is a weight, $\max(\boldsymbol{x}_s, \boldsymbol{x}_t)$is the vector with each element obtained by taking the maximum among the corresponding element of $\boldsymbol{x}_s$ and $\boldsymbol{x}_t$. For example, $\max([1\ \ -2\ \ 3], [2\ \ 3\ \ 1]) = [2\ \ 3\ \ 3]$. Similarly, $\min(\boldsymbol{x}_s, \boldsymbol{x}_t)$ denotes a vector obtained by taking the minimum value. Among these 4 chromosomes, the two with superior fitness value are taken as the offspring of the crossover operation. It can be seen that the potential offspring can spread over the domain, and this crossover operator is superior to the single arithmetic crossover or heuristic crossover.

3) **Adaptive change of mutation scaling**

Different from the uniform mutation, boundary mutation, *etc*, a mutation operator with adaptive change of mutation scaling is employed. According to "the punctuated equilibrium theory"[6] in the evolution field, species evolution always appears in many and different directions at previous stage while the evolution tends to be conservative at later stage. Therefore, a larger mutation range is employed during previous stage to keep the population diversified while the mutation range will be shrunken gradually during later stage to focus on local search.

Supposing that the mutation scaling is μ $(0 \leqslant \mu \leqslant 1)$, the element x_k $(x_k \in [x_k^{\min}, x_k^{\max}])$ selected in the chromosome $(x_1, x_2, \cdots, x_k, \cdots, x_n)$ is to be mutated with a certain mutation probability Pm. Then this original value x_k will be replaced by the mutated value x_k^{new} chosen from the range

$$x_k^{new} \in \Big\{\max(x_k - \mu\tfrac{x_k^{\max} - x_k^{\min}}{2}, x_k^{\min})\ , \\ \min(x_k + \mu\tfrac{x_k^{\max} - x_k^{\min}}{2}, x_k^{\max})\Big\} \tag{9}$$

with a uniform probability.Based on the concept that the mutation scaling μ is decreasing gradually during the process, a monotonously decreasing function of the mutation scaling μ is built

$$\mu(\tau) = 1 - r^{(1-\frac{\tau}{T})^b} \tag{10}$$

where T is the number of generations, τ is the current iteration, and the weight $r \in [0,1]$. From the formula (10) it can be seen that for a small value of weight r, the mutation scaling μ is near to one at the beginning of the optimization, and the mutation scaling μ will be decreasing down to zero as the run progresses.

4) **Adaptive strategy of stage evolution**

During the process of evolution, the diversity of population is descending. When the diversity decreases to a certain level, the algorithm searching is over[1]. Generally, at a previous stage larger crossover and mutation probability can work obviously, while at a later stage the crossover and mutation probability had better be smaller since the algorithm has entered into the local searching process. For the selection operator, it is a good idea to choose smaller selection pressure at the beginning, and adopt larger selection pressure later to promote local searching.

Based on this idea, a model based on stage evolution is developed. We divide the whole process into 3 stages:

First stage:	$\tau \in [0, T_1]$	$T_1 = \alpha T$
Second stage:	$\tau \in (T_1, T_2]$	$T_2 = (1-\alpha)T$
Third stage:	$\tau \in (T_2, T]$	

where T and τ have been defined as above, generally parameter α is equal to 0.382. Then we choose three different best individual's selection probability q, crossover probability Pc and mutation probability Pm for each stage.

2.2 Hybrid simplex-improved genetic algorithm (HSIGA)

In order to improve the local fine tuning capability of GA and quicken the convergence rate, we combine the IGA with the simplex method (SM) to form a hybrid optimization algorithm[7]. The detailed process is as follows.

All chromosomes in the current generation are arranged from the best to the worst firstly, and new population in the next generation is generated through the following three approaches.

1) Elitist strategy: The first N top-ranking chromosomes (elites) are reproduced directly into the next generation so that these elites can not be destroyed by the 3 operations of the GA or other operations.

2) Modified simplex strategy: In this HSIGA algorithm, the S $(S > N)$ top members in population produce S-N new chromosomes through the modified simplex method. In modified simplex method, the new generated chromosome is generated by reflecting $\boldsymbol{x}_j$ over the centroid $\boldsymbol{x}_c$ of the remaining points as follows.

$$\boldsymbol{x}_j^{new} = \boldsymbol{x}_c + \alpha(\boldsymbol{x}_c - \boldsymbol{x}_j) \quad (j = N+1, \cdots, S) \tag{11}$$

where the centroid is equal to $\boldsymbol{x}_c = (\boldsymbol{x}_1 + \boldsymbol{x}_2 + \cdots + \boldsymbol{x}_N)/N$, α is a random value.

3) Improved genetic algorithm (IGA) strategy: The remaining P-S children (where P is the population size) in the new generation are created through the IGA acting on the whole population.

Fig.1 depicts the architecture of this HSIGA algorithm. We can refer to the hybrid degree (S/P) by using the percentage of population to which the modified simplex operator is applied. From it we can see that the hybrid algorithm will become a real-code IGA when the hybrid degree (S/P) is zero; while the hybrid degree (S/P) is equal to 100%, the algorithm will turn into the modified simplex method. Generally S is around 20 percent of the size P.

```
begin
    $t \to 0$    //$t$: number of iterations
    initialize $P(t)$    //$P(t)$: population for iteration $t$
    evaluate $f(P(t))$    //$f(P(t))$: fitness function
while (not termination condition) do
  begin
    $t \to t + 1$
    rank the population according to the fitness
    produce a new population through the following 3 approaches
      (Elites Strategy) copy $N$ elitists directly into the next population
      (Modified Simplex Strategy) create $S$-$N$ new chromosomes from the former $S$ excellent chromosomes
      (IGA Strategy) create $P$-$S$ new chromosomes through the IGA acting on the whole population
    evaluate $f(P(t))$
  end
end
```

Fig.2 Procedure of the HSIGA algorithm

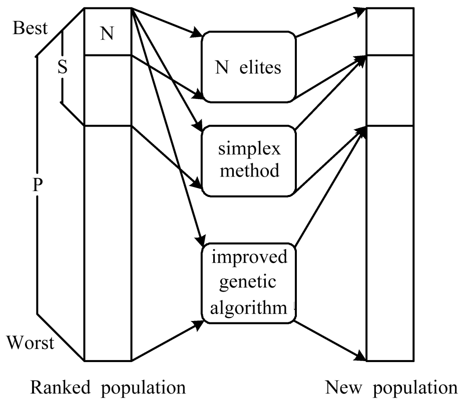

Fig.1 Architecture of the hybrid simplex-IGA algorithm

The new population in HSIGA is produced through these 3 strategies with the following advantages.

1) In GA the coding of elitists would be changed or destroyed after the genetic operation, and the produced offspring may be less fitness than their parents. Elitist strategy is an effective approach to avoid the damage of the elitists, which is necessary to enhance the capacity of the algorithm.

2) Some novel genetic operations are used in IGA, such as the crossover and the mutation operator. The idea of these operations is mainly from the nature. The genetic mechanisms try to mimic the maturing phenomenon in nature, which makes the individuals more suitable to the environment, and enhances the optimization performance.

3) GA is global search procedure. It is less likely to be trapped in local optima, but the convergence rate will slow down and the computational cost is high at later stage. SM is a local search method whose merits are simple and computationally efficient, however it is easily entrapped in a local optimum. The hybrid of these two methods can combine the respective merits, which can speed up the convergence rate and avoid being entrapped in a local optimum. Moreover, HSIGA applies simplex reproduction to the top-ranking individuals, and applies simplex search to many points not a single one in the vicinity of optimum, which can quicken the convergence rate greatly.

Based on the above, the pseudo code of this HSIGA algorithm can be depicted in Fig.2.

3 Numerical experiment and results

We execute the HSIGA to solve 10 benchmark functions[8~12] in Table 1. f_1-f_6 are multimodal functions where the number of local minima increases with the problem dimension; f_7-f_{10} are unimodal functions.

In some recent studies, functions f_1-f_{10} were tested by the FEP[8], OGA/Q[9], CEP[10], PSO[11], EO[11], and FES[12] optimization algorithms. From the results of these numerical experiments, it can be seen that the OGA/Q[9] can give more robust and significantly better results than some other optimization algorithms, such as the CEP[10], PSO[11], EO[11], FES[12] *etc.* Therefore we adopt the ten benchmark functions to test our proposed HSIGA, and only to compare the performance of the HSIGA with the performance of FEP[8] and OGA/Q[9].

3.1 Control experiment

In order to identify any improvement due to improved genetic operations and combination with the modified simplex method, the following control experiments are designed and carried out. We execute the IGA and SGA to solve the test functions, where IGA is similar to HSIGA basically, except for the top members' number S in the population. In IGA the number of top members S is equal to zero, while in HSIGA S is 20 percent of the size P. SGA neither apply any improved genetic mechanisms nor combines with other optimization method; it adopts only traditional operations, such as fitness-proportionate selection, arithmetic crossover and uniform mutation operation.

3.2 Parameter values

Before solving these test functions, some parameters should be assigned for each algorithm.

Table 1 List of 10 test functions used in experimental study, where n is the function dimension, n=30

Test functions	Solution space
$f_1(x) = \sum_{i=1}^{n} (-x_i \sin\sqrt{\lvert x_i \rvert})$	$[-500, 500]^n$
$f_2(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^n$
$f_3(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + \exp(1)$	$[-32, 32]^n$
$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]^n$
$f_5(x) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \leqslant x_i \leqslant a; \quad y_i = 1 + \frac{1}{4}(x_i + 1) \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-50, 50]^n$
$f_6(x) = \frac{1}{10}\left\{\sin^3(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50, 50]^n$
$f_7(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$
$f_8(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]^n$
$f_9(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	$[-100, 100]^n$
$f_{10}(x) = \max\{\lvert x_i \rvert,\ i = 1, 2, \cdots, n\}$	$[-100, 100]^n$

For the HSIGA and the IGA algorithms:

• Population size: We choose a moderate population size $P = 60$.

• Genetic probabilities: The whole evolution is divided into 3 stages. Table 2 shows the best chromosome's selection probability q, crossover probability Pc and mutation probability Pm at each stage.

• Mutation parameters: The parameter r in the mutation scaling function is 0.5, and parameter b is equal to 2.

• Stopping criterion: For each function has different complexity, we use different stopping criterion. Table 3 lists the evolution generations T for each function. When the current iteration τ reaches T, the execution is stopped.

In addition, the modified simplex strategy is adopted in the HSIGA algorithm.We choose the hybrid degree of the HSIGA algorithm (S/P)=20%, and elites N is equal to four. In IGA the hybrid degree is equal to zero.

For the SGA algorithm:

• Population size: Population size of the SGA is 150, which is larger than that of the IGA and the HSIGA.

• Selection operation: Fitness-proportionate selection is adopted.

• Crossover operation: Arithmetic crossover is employed and crossover probability Pc is 0.80.

• Mutation operation: Uniform mutation is used and mutation probability Pm is 0.02.

• Stopping criterion: In SGA if the fitness value of the best chromosomes cannot be further reduced in successive 50 generations after 500 generations, the execution of the algorithm is stopped.

3.3 Results and comparison

Since the SGA uses different evolutionary parameters and termination criterion from those adopted by the IGA and the HSIGA, to make a fair comparison, we will calculate firstly the computational cost of each algorithm, and compare the qualities of their final solutions at the given computational cost.

Each test function is performed in 50 independent runs and the following results are recorded: 1) the mean number of function evaluations; 2) the mean function value; and 3) the standard deviation of the function values. Table 4 shows each algorithm's results in the control experiment. From these results in Table 4 it can be seen that

• As can be seen, HSIGA finds the exact global optimum, 0, for functions f_2, f_4 and $f_7 \sim f_{10}$. For other functions the mean function values of HSIGA are close to the optimal ones and the standard deviations are small. These results indicate that HSIGA can find optimal or close-to-optimal solutions, and its solution quality is quite stable.

• Compared to the results of SGA, the proposed HSIGA algorithm requires less function evaluations than SGA, and hence it has a smaller time complexity.However, HSIGA gives significantly smaller and closer-to-optimal solution than SGA, and hence its mean solution quality is much better than SGA. In addition, HSIGA gives smaller standard deviations of function values than SGA, so its solution quality is more stable.

• Compared to the results of IGA, though the generations T of these two algorithms are equal, HSIGA requires less function evaluations than IGA. This is because part individuals of next generation in HSIGA are produced through elite strategy and modified simplex strategy. However, HSIGA gives smaller mean function values, and gives equal or smaller standard deviations of function values than IGA for the 10 functions, and hence the solution quality of HSIGA is better than that of IGA, and the HSIGA algorithm is statistically stable.

Next, the performance of HSIGA is compared with the that of FEP[8] and OGA/Q[9] algorithms. Since in [8] and [9] the optimization results using these two algorithms are available for the 10 test functions, the comparison will be made accordingly and the results are listed in Table 5.

Table 2 Adaptive change value of parameter (τ: Current iteration; T: Evolution generations)

τ	q	Pc	Pm
$\tau \in [0, T_1]$	0.08	0.95	0.08
$\tau \in (T_1, T_2]$	0.10	0.80	0.05
$\tau \in (T_2, T]$	0.12	0.65	0.02

Table 3 HSIGA and IGA number of generations for each function (**TF**: Test function **NG**: Number of generations)

TF	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
NG	500	30	50	40	500	500	400	500	400	500

Table 4 Comparison of optimization result and computation effort among HSIGA, IGA and SGA

f	Mean number of function evaluations			Mean function value (standard deviation)			f_{min}
	HSIGA	IGA	SGA	HSIGA	IGA	SGA	
f_1	68,412	78,052	89,877	-12569.4740 (1.0604E-4)	-12569.4530 (1.3782E-2)	-6422.1504 (869.3862)	-12569.5
f_2	4,130	4,698	259,965	0 (0)	7.0237E-14 (2.4246E-13)	1.9255 (8.0727)	0
f_3	6,853	7,825	229,926	8.8818E-16 (0)	5.8950E-13 (2.1217E-12)	6.2034E-2 (3.4579E-2)	0
f_4	5,482	6,275	146,463	0 (0)	2.2693E-15 (6.9245E-15)	7.0871E-1 (3.8481E-1)	0
f_5	68,429	78,044	319,695	3.2475E-6 (3.5963E-6)	1.7855E-5 (9.0217E-6)	3.5947E-4 (6.2180E-4)	0
f_6	68,417	78,044	334,896	9.0181E-5 (1.0448E-5)	5.0194E-4 (1.5354E-3)	1.0965E-2 (2.2325E-2)	0
f_7	54,745	62,446	322,368	0 (0)	2.2817E-183 (0)	9.6696E-2 (8.4127E-2)	0
f_8	68,434	78,002	211,680	0 (0)	8.9611E-115 (5.7284E-114)	3.8576E-2 (2.0601E-2)	0
f_9	54,702	62,391	269,658	0 (0)	5.3675E-186 (0)	219.8387 (118.2618)	0
f_{10}	68,361	78,004	126,852	0 (0)	6.3391E-116 (3.1639E-115)	9.0751E-1 (2.6103E-1)	0

Table 5 Comparison between HSIGA and OGA/Q[9] and FEP[8]

f	Mean number of function evaluations			Mean function value (standard deviation)			f_{min}
	HSIGA	OGA/Q[9]	FEP[8]	HSIGA	OGA/Q[9]	FEP[8]	
f_1	68,412	302,166	900,000	-12569.4740 (1.0604E-4)	-12569.4537 (6.4470E-4)	-12554.5 (52.6)	-12569.5
f_2	4,130	224,710	500,000	0 (0)	0 (0)	4.6E-2 (1.2E-2)	0
f_3	6,853	112,421	150,000	8.8818E-16 (0)	4.4400E-16 (3.9890E-17)	1.8E-2 (2.1E-3)	0
f_4	5,482	134,000	200,000	0 (0)	0 (0)	1.6E-2 (2.2E-2)	0
f_5	68,429	134,556	150,000	3.2475E-6 (3.5963E-6)	6.0190E-6 (1.1590E-6)	9.2E-6 (3.6E-6)	0
f_6	68,417	134,143	150,000	9.0181E-5 (1.0448E-5)	1.8690E-4 (2.6150E-5)	1.6E-4 (7.3E-5)	0
f_7	54,745	112,559	150,000	0 (0)	0 (0)	5.7E-4 (1.3E-4)	0
f_8	68,434	112,612	200,000	0 (0)	0 (0)	8.1E-3 (7.7E-4)	0
f_9	54,702	112,576	500,000	0 (0)	0 (0)	1.6E-2 (1.4E-2)	0
f_{10}	68,361	112,893	500,000	0 (0)	0 (0)	0.3 (0.5)	0

From Table 5, it can be seen that the HSIGA exhibits more accurate results and a superior performance over FEP, while the required computational effort is less than that required by FEP for all the functions. For f_1, f_5, and f_6 HSIGA can give smaller mean function values using smaller numbers of function evaluations than OGA/Q. For f_2, f_4 and $f_7 \sim f_{10}$, the solutions of HSIGA are as good as those of OGA/Q, but the mean number of function evaluations of HSIGA is much lower than those of OGA/Q. For function f_3, HSIGA gives a mean function value of 8.8818E-16, which is already very close to the global minimum 0, using 6 853 function evaluations; OGA/Q gives a mean function value 4.4400E-16, which is smaller than that of HSIGA, but it uses 112 421 function evaluations, namely, 16 times the computational cost of HSIGA. In other words, HSIGA can find close-to-optimal solutions, but OGA/Q can find closer-to-optimal solutions using much more function evaluations for function f_3. In addition, HSIGA gives smaller standard deviation of function values than FEP and gives equal or smaller standard deviation of function values than OGA/Q (except function f_5), hence it has a more stable solution quality. To summarize, the results show that HSIGA outperforms FEP and OGA/Q, and is competent for the numerical optimization problems.

4 Conclusion

In this paper, the HSIGA has been presented to solve global numerical optimization problems.This methodology involves a novel improvement on the genetic mechanism and combination with the modified simplex method to enhance the genetic algorithm.The HSIGA has been carried out to solve 10 benchmark problems with high dimensions. Results obtained from 50 trials for each function show that the proposed HSIGA is able to find optimal or close-to-optimal solutions for all these test functions; moreover the behavior of the algorithm is stable. Comparison of the HSIGA outcome with those from several other global optimization algorithms demonstrates that the HSIGA is more competitive than some recent algorithms on the problem studied.

References

1 Li Min-Qiang, Kou Ji-Song, Lin Dan, Li Shu-Quan. *Genetic Algorithm Basic Theory and Application*. Beijing: Science Press, 2002 (in Chinese)

2 Leung Frank H F, Lam H K, Ling S H, Tam Peter K S. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks*, 2003, **14**(1): 79∼88

3 Fogel D B . *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE, 2005.

4 Sareni B, Krahenbuhl L. Fitness sharing and niching methods revisited. *IEEE Transactions on Evolutionary Computation*, 1998, **2**(3): 97∼106

5 Wang Deng-Gang, Liu Ying-Xi, Li Shou-Chen. Hybrid genetic algorithms of global optimum for optimization problems. *Acta Mechanica Sinica*, 2002, **34**(3): 469∼474 (in Chinese)

6 Zhang Yun. Rate of evolution and unification of evolution theories. *Acta Scientiarum Naturalium Universitatis Pekinensis*, 1997, **33**(6): 794∼803 (in Chinese)

7 Yen J, Liao J C, Lee B, Randolph D. A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 1998, **28**(2): 173∼191

8 Yao X, Liu Y, Lin G M. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 1999,**3**(2): 82∼102

9 Leung Y W, Wang Y P. An orthogonal genetic algorithm with quantization for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 2001,**5**(1): 41∼53

10 Chellapilla K. Combining mutation operators in evolutionary programming. *IEEE Transactions on Evolutionary Computation*, 1998,**2**(3): 91∼96

11 Angeline P J. Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In: *Proceeding of Evolutionary Programming VII Lecture Notes in Computer Science.* **1447**, Berlin, Germany: Springer-Verlag, 1998, 601∼610

12 Yao X, Liu Y, Lin G M. Fast evolution strategies. In: *Proceedings of Evolutionary Programming VI Lecture Notes in Computer Science.* **1213**, Berlin, Germany: Springer-Verlag, 1997, 149∼161

REN Zi-Wu Ph. D. candidate in Control & Simulation Centre at Harbin Institute of Technology, China. His research interests include evolutionary algorithms and neural networks. Corresponding author of this paper. E-mail: zwrenjren@yahoo.com.cn.

SAN Ye Professor in Control & Simulation Centre at Harbin Institute of Technology, China. His research interests include system simulation techniques and optimal control.

CHEN Jun-Feng Assistant in College of Computer & Information Engineering at Hohai University, China. Her research interests include genetic algorithm (GA) and neural network control.