2012 IEEE International Conference on Systems, Man, and Cybernetics
October 14-17, 2012, COEX, Seoul, Korea

Guided Mutation Operation Based on Search Degree and Fitness Estimation

Liang Yin and Jun Zhang (Corresponding Author)
Department of Computer Science, Sun Yat-sen University
Key Laboratory of Machine Intelligence and Sensor Network, Ministry of Education
Key Laboratory of Software Technology, Education Dept. of Guangdong Province, P.R. China
junzhang@ieee.org

Abstract—**Mutation is a fundamental operation in genetic algorithm (GA) for it has a significant impact on global search ability and convergence rate. Traditional mutation operation of GA changes chromosomes randomly (or blindly), which would waste a lot of computational cost in searching less promising regions or those have been searched frequently. To address these drawbacks, this paper proposes a novel guide mutation. The proposed guide mutation makes use of history search experience to estimate the average fitness and search degree of sub-regions in the search space. New chromosomes generated by the guide mutation are more likely to be in regions with higher average fitness and less search degree. In this way, the search efficiency can be improved and the algorithm can have a stronger ability of jumping out of local optima. The proposed guide mutation is incorporated into a simple GA, forming a guided mutation GA (GMGA). The GMGA is validated by testing 23 benchmark functions and the experimental results reveal that the proposed guide mutation is very effective in improving the performance of GA.**

I. INTRODUCTION

Genetic algorithm (GA) which was first proposed by J. Holland at 1975 [1], is a notable stochastic optimization algorithm inspired by the natural biological evolution mechanism. It starts with a randomly initialized population. Then new individuals are generated by performing crossover and mutation on the current population, and better individuals are selected to form the new population. Due to its simple mechanism and easy implementation, GA, as well as ant colony algorithm and discrete PSO[2], has been successfully applied to a wide range of applications such as machine learning, workflow scheduling and wireless sensor network, etc [3] [4] [5][6].

Though GA performs well in a number of problems, its capability still has great room for improvement. The mutation operation is one of the hot research spot. In [7], De Jong suggested that the mutation rate should be 0.001, so as to balance the exploit and explosive of GA. In [8], however, J. D. Schaffer et al. suggested that [0.005, 0.01] is an appropriate range of mutation rate. In [9], T. Back suggested that the mutation rate should decrease as the generation increases. As well as PSO [10], some scholars also proposed and improved a self-adapt mechanism to control the mutation rate of GA [11] [12] [13]. Besides controlling the mutation rate, various efforts have been made to change the mutation mechanism to improve GA's efficiency. In [14] [15], Schwefel H P et al. presented a popular gauss mutation operation based on Gauss random distribution. Later, Cauchy random distribution is also utilized to improve the performance of mutation operation [16], [17]. To improve the global search ability of GA, Gao Ye et al. [18] combined chaotic algorithm with mutation operation and then use this enhanced mutation operation in rout path for robot.

Although existing mutation mechanisms are effective, there are some drawbacks. First, they blindly change individuals without any guidance. This may cause the algorithm wasting a lot of computational cost in search regions which are frequently searched. Second, they are not efficient enough because they usually shift chromosomes to regions which have little potential to contain the global optimal chromosome. In order to overcome the above shortages, this paper proposed a novel guide mutation. The proposed guide mutation is incorporated into a simple GA and form a guided mutation GA (GMGA). In GMGA, all chromosomes found during the evolution are preserved and are utilized to estimate the average fitness and search degree of each region. The average fitness is used to evaluate the potential of the region containing the global optimal chromosome, while the search degree is used to evaluate the frequency of the region being searched. Based on the average fitness and search degree, the guide mutation is more likely to generate new individuals in regions with higher average fitness and lower search degree. To validate the proposed guide mutation, extensive experiments are conducted and the results show that the guide mutation can significantly improve the performance of GA

In the next section, this paper makes a brief review of GA. section III presents the detail of implementation of the guide mutation. The experiments and comparison are presented in Section IV, followed by the conclusion in Section V.

II. BREIEF REVIEW OF GENETIC ALGORITHM

Simple genetic algorithm (SGA) contains 5 steps.

Step 1: Initialize operation:

SGA starts with initialing the parameters. There are five parameters in SGA: maximum number of generations (*MAXGEN*), Population size (*PS*), mutation rate (*PM*), possibility of crossover (*PC*) and gene number of

This work was supported in part by the National Science Fund for Distinguished Young Scholars No.61125205, National Natural Science Foundation of China No.61070004 and NSFC Joint Fund with Guangdong under Key Project U0835002.

978-1-4673-1714-6/12/$31.00 ©2012 IEEE

chromosomes (*GN*). After initialing these parameters, SGA randomly generates a population.

Step 2: Select operation:

The aim of the select operation is choosing fitter chromosomes into the next generation. In order to choose fitter chromosomes, SGA applies tournament operation [19]: randomly select 3 chromosomes, compare their fitness values, and select the highest one into the new population.

Step 3: Crossover operation.

In this operation, SGA selects two chromosomes, randomly generates a number between one and *GN* as the crossover point, and then swaps the genes before the crossover point.

Step 4: Mutation operation.

In this operation, SGA randomly generates a number within the boundaries to replace the gene involved in the mutation operation.

Step 5: Elitist operation.

In the elitist operation, SGA preserves a global best chromosome (*GB*) —the best chromosome has ever been found. If the population's best chromosome that is fitter than *GB*, the population's best chromosome will replace *GB*. Else, *GB* will replace the population's worst chromosome. Afterward, the algorithm goes to step 2 and begins the next generation.

III. Guided Mutation Operation Based on Search Degree and Fitness Estimation

The single-object problem can be described as equations (1) and (2):

$$\min f(x^1, x^2 \ldots x^{GN}); \tag{1}$$

$$l^i \le x^i \le u^i; \tag{2}$$

Vector $x=\{x^1,x^2,\ldots x^{GN}\}$ is a chromosome and x^i is the i^{th} gene of x. $[l^i,u^i]$ is the boundary of x's i^{th} gene. $P=\{p_1,p_2\ldots p_n\}$ is the set of all evaluated chromosomes, where n is the number of evaluated chromosomes. $p_k(k=1,2\ldots n)=\{p_k^1,p_k^2,\ldots p_k^{GN}\}$, The i^{th} gene of p_k is p_k^i.

Based on these concepts, this paper introduces guided mutation operation in the following part.

A. Motivation

Mutation operation has significant impacts on the convergence rate and the global search ability of GA. However, the mutation operation which shifts a chromosome randomly (or blindly) has two drawbacks.

The first drawback is that blind mutation operation may cause plenty of regions ' getting insufficient search while some regions are over-searched. To solve this problem, it is desirable to make use of history search experience to estimate a region's search degree and use this information to guide the mutation.

The other drawback is that blind mutation operation might shift a chromosome to a region without potential to find the global optimum. Assuming that the global optimum is more likely to be in regions with higher fitness, it is desirable to utilize the history search experience to estimate regions' average fitness and use this information to guide the mutation.

With regards of the above two factors, we proposed a guided mutation operation, aiming to lead chromosomes jump into a region with higher potential and been less-searched.

B. The Proposed Guide Mutation

If the i^{th} gene of a chromosome needs to mutate, there are 7 steps:

Step 1: equally divide the boundary of the chromosome's i^{th} gene into *PN* segments. Thus, the whole search space has been divided into *PN* regions by its i^{th} dimension. A set of regions $s_j(j=1,2\ldots PN)$ is formulated and the i^{th} gene's boundary of $s_j(j=1,2\ldots PN)$ is $[l_j^i,u_j^i]$.

$$l_j^i = l^i + \frac{(u^i - l^i)}{PN} * (j-1); \tag{3}$$

$$u_j^i = l^i + \frac{u^i - l^i)}{PN} * j; \tag{4}$$

Step 2: estimate regions' search degree (*sd*), shown in equation (5) and equation (6).

$$sd_j = \sum_{k=1}^{n} g_j(p_k); \tag{5}$$

$$g_j(p_k) = \begin{cases} 1; & if \ \ l_j^i < p_k^i \le u_j^i \\ 0; & else \end{cases} \tag{6}$$

Where sd_j is the search degree of $s_j(j=1,2\ldots PN)$. If equals x, it means region, s_j contains x evaluated chromosomes.

Step 3: estimate the average fitness value of regions (*af*). Equation (7) shows how to calculate x's fitness, where x is a chromosome:

$$eval(x) = \frac{1}{(f(x) - f_{\min})}; \tag{7}$$

Where $eval(x)$ is the fitness value of chromosome x, and f_{min} is the global minimum. Base on (7), equation (8) and (9) shows how to estimate s_j's average fitness value af_j :

$$af_j = \frac{\sum_{k=1}^{n} h_j(p_k)}{sd_j}; \tag{8}$$

$$h_j(p_k) = \begin{cases} eval(p_k); & if \ \ l_j^i < p_k^i \le u_j^i \\ 0; & else \end{cases} \tag{9}$$

Step 4: calculate Guide Information (*gi*) of $s_j(j=1,2\ldots PN)$ according to sd_j and af_j. *gi* is used for guiding the mutation operation. As this paper mentioned before, guided mutation operation is more likely to lead a chromosome into a region with high fitness and that has been less-searched. Obviously, Region's search degree could be reflected by sd_j, the larger the sd_j, the more the region has been searched. Thus, a segment with small sd_j and large af_j is competitive. The equation calculates *gi* of $s_j(j=1,2\ldots PN)$ is described as follow:

$$gi_j = \frac{af_j}{sd_j}; \tag{10}$$

Step 5: select a fitter region to guided mutation operation. In this step, GMGA randomly choose 3 regions from *PN*

regions, compare their *gi* and Choose the region with the largest *gi* to next step.

Step 6: Suppose $s_j(j=1,2...PN)$ is the selected region, the algorithm randomly generate a number within $[l_j^i, u_j^i]$ and use this number to replace the chromosome's i^{th} gene.

Step 7: Update the set *P*. For new chromosomes that have been generated via mutation operation, this new chromosome should be added into the set *P* as well as adjust *n*.

Since the set $P=\{p_1 p_2 ... p_n\}$ preserves all evaluated chromosomes, it should be updated when a new chromosome has been generated. Thus, crossover operation should update the set *P* as well as the mutation operation.

IV. Experiment and Comparison

A. Experiment Setup

In this section, 23 benchmark functions are used for comparing the performance of GMGA and GA in the experiment. In these functions, functions 1-7 are unimodal functions and functions 8-13 are high-dimensional multimodal functions with plenty of local minima and functions 14-23 are low-dimensional multimodal functions with a few local minima. These functions and their detail information are showed in [20].

Parameters of GMGA and SGA used in functions 1-23 are listed in TABLE I. [20]

TABLE I. Where PM is the Mutation Rate, PC is the Possibility of Crossover, PN is the Partition Numgber, PS is the Population Size

Algorithm	PM(1-13)	PM(14-23)	PC	PN	PS
GMGA	0.01	0.25	0.8	200	100
GA	0.01	0.25	0.8	--------	100

The operation environment of this experiment is: Intel(R) core(TM) i3 CPU. M380 @ 2.53GHZ, 2.53GHZ, Ram 2.00GB, visual C++ 6.0, WINDOWS 7 operation system. Every function runs 30 times independently and takes the average result.

B. Comparison Between GMGA and GA

TABLE II shows the performance of GMGA and GA in functions 1-7, and convergence process of these functions are shown in Fig. 1:

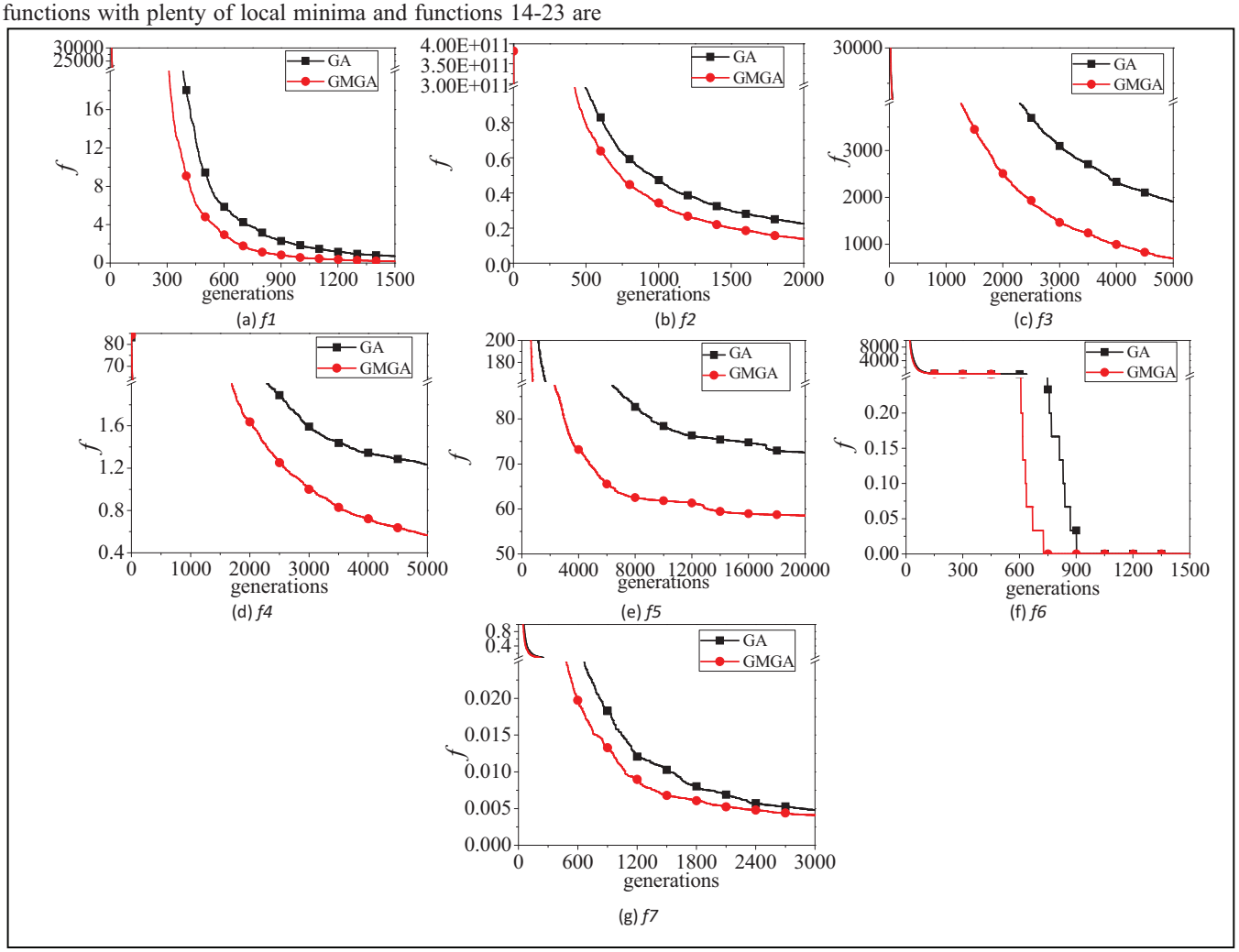

Figure 1. Convergence process of GMGA and GA in functions 1-7.

TABLE II. WHERE MEAN IS THE AVERAGE RESULTS IN 30 TIMES, STDDEV IS THE SQUARE DEVIATION.

Function	*MAXGEN*	GMGA		SGA	
		Mean	Stddev	Mean	Stddev
1	1500	**0.177977**	$4.5*10^{-3}$	0.710882	0.06
2	2000	**0.138960**	$8*10^{-4}$	0.225369	$1*10^{-3}$
3	5000	**693.44**	46965	1911.14	564257
4	5000	**0.565117**	$8.6*10^{-3}$	1.233762	0.16
5	20000	**58.543**	1141	72.541088	887
6	1500	**0.000000**	0	0.000000	0
7	3000	**0.004121**	$1.2*10^{-6}$	0.004788	$2.3*10^{-6}$

As this paper mentioned before, Functions 1-7 are unimodal functions. Thus, these functions focus on comparing the convergence rate of GMGA to SGA for they are difficult to track into local optimum. It is obvious that the convergence rate of GMGA is much faster than SGA for guided mutation operation plays an important role in improving convergence rate. Because of the lack of local search operation in SGA and GMGA, the final outcomes may not be as good as many other algorithms.

In functions 8-13, especially in function 8, although the final results of GMGA are better than GA, the convergence rate of GMGA is slow at first. The performance of GMGA and SGA in functions 8-13 are showed in TABLE III. The convergence process of GMGA and SGA in functions 8-13 are showed in Fig. 2:

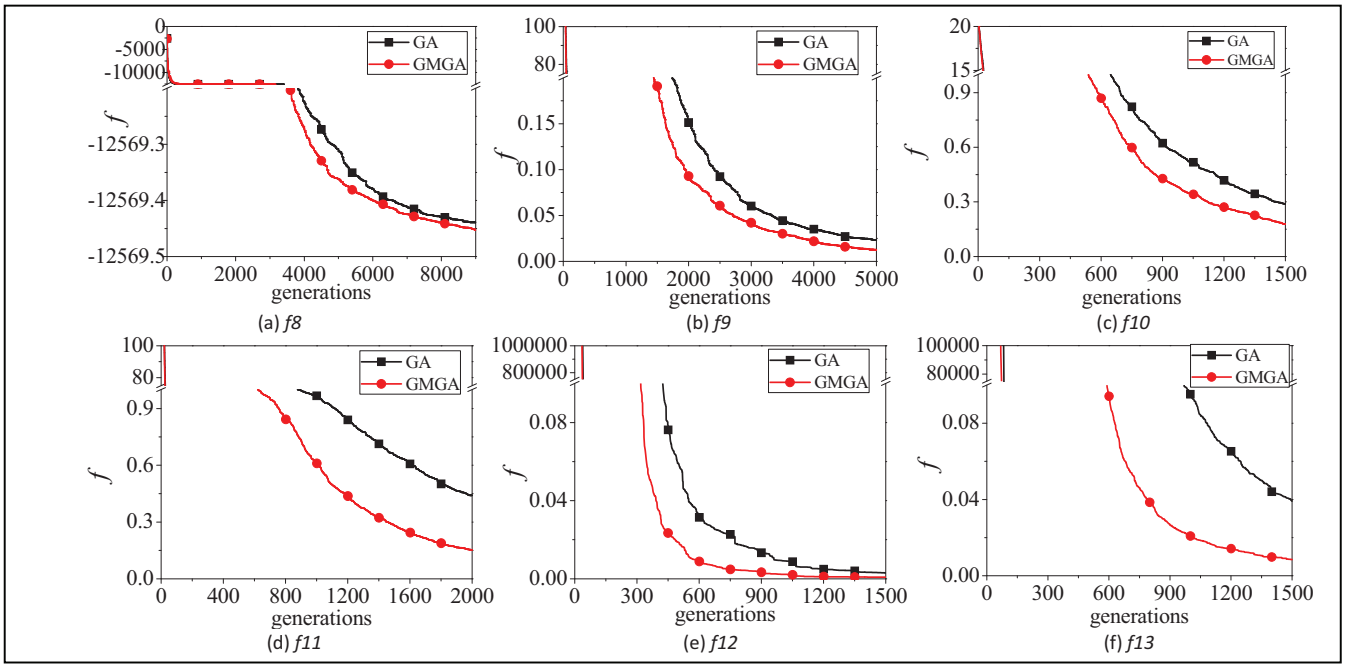

Figure 2. Convergence process of GMGA and GA in functions 8-13.

TABLE III. WHERE MEAN IS THE AVERAGE RESULTS IN 30 TIMES, STDDEV IS THE SQUARE DEVIATION.

Function	*MAXGEN*	GMGA		SGA	
		Mean	Stddev	Mean	Stddev
8	9000	**-12569.453**	$1.8*10^{-4}$	-12569.450	$2.8*10^{-4}$
9	5000	**0.012383**	$3.6*10^{-5}$	0.023190	$1*10^{-4}$
10	1500	**0.178854**	$1.4*10^{-3}$	0.289474	$3*10^{-3}$
11	2000	**0.152551**	$1.4*10^{-3}$	0.441348	0.014
12	1500	**0.000854**	$8.4*10^{-7}$	0.003043	$3.9*10^{-6}$
13	1500	**0.008497**	$3*10^{-5}$	0.039171	$2.3*10^{-4}$

Functions 8-13 are multimodal functions with plenty of local minima and high dimensions. Thus, plenty of regions contain high fitness chromosomes. In the beginning, the estimated chromosomes are so limited that can't accurately reflect the average fitness of a region, therefore, GMGA regards a region containing high fitness chromosome as the best region. After a great number of chromosomes have been considered, the accuracy of regions' fitness is much more precise than before. Thus, the region contains global optimum can be found and the final outcomes of GMGA are much better than SGA. Guided mutation operation improves GA's global search ability and the accuracy in high dimension functions. GMGA and SGA both are with great global search ability for they can make large jumps in mutation operation. Therefore, they can obtain good outcomes in the end. However, due to the lack of local search ability, they hardly obtain the global optimum.

Though functions 14-23 are also multimodal functions, they are with low dimensions and a few local minima. The performance of GMGA and GA in functions 14-23 are listed in TABLE IV, the convergence process are showed in Fig. 3:

It is obvious that GMGA is not much better than SGA in these functions. In functions 21-23, GMGA is unstable. Take 23 for example, the global optimum of function 23 is -10, this paper runs GMGA for 1000 times, the possibility of the result being less than -9 is nearly 61%. There are two reasons contribute to this situation. One is that GMGA only runs 100 generations, the region with global optimum is difficult to be

TABLE IV. WHERE MEAN IS THE AVERAGE RESULTS IN 30 TIMES, STDDEV IS THE SQUARE DEVIATION.

Function	*MAXGEN*	GMGA		SGA	
		Mean	Stddev	Mean	Stddev
14	100	**1.000003**	$4.6*10^{-13}$	1.000021	$3.3*10^{-9}$
15	4000	**0.000558**	$3.8*10^{-9}$	0.000590	$9.8*10^{-9}$
16	100	**-1.031320**	$5.3*10^{-9}$	-1.031096	$1.5*10^{-7}$
17	100	**0.398055**	$2.3*10^{-7}$	0.398100	$5.7*10^{-8}$
18	100	**3.000350**	$1.2*10^{-6}$	3.005933	$9.2*10^{-5}$
19	100	-3.723845	$2.7*10^{-6}$	**-3.724530**	$2.7*10^{-6}$
20	200	-3.243747	$2.2*10^{-3}$	**-3.257210**	0.03
21	100	**-8.023451**	5.9	-6.663288	7.3
22	100	-8.083881	2.8	**-8.513930**	5.4
23	100	**-9.085124**	4.1	-8.933987	5.8

discovered. The other reason is that GMGA lacks local search ability. Although the region with global optimum is found, it is hard for GMGA to find a great result in a large region.

C. Parameters Analysis

GMGA introduces a new parameter, partition numbers (*PN*). This section discusses the relationship between *PN* and GMGA's performance. Fig. 4 shows the performance of GMGA with different *PN* in function 1.

GMGA performs great from 20 -200. However, from 200 to 500 the performance is getting worse with the increase of *PN*. This situation may account for the small competition

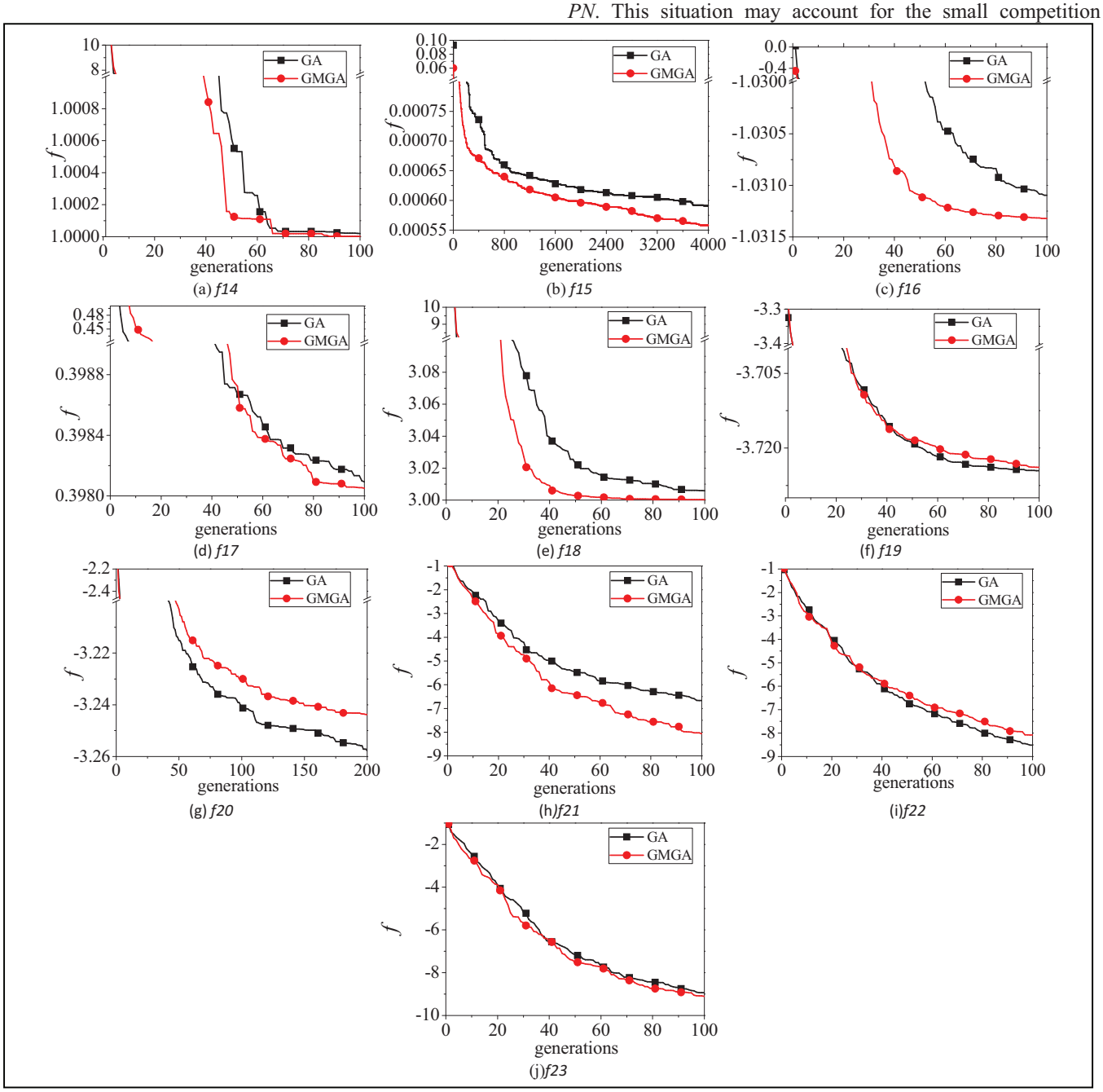

Figure 3. Convergence process of GMGA and GA in functions 14-21

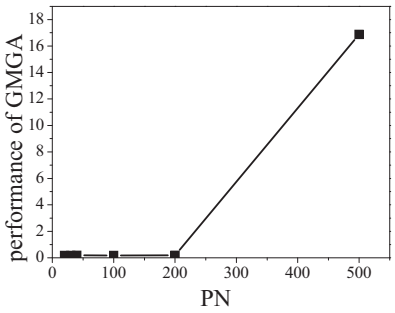

Figure 4. Relationship between PN and performance of GMGA in function1.

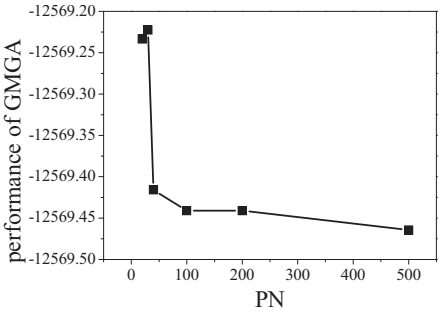

Figure 5. Relationship between PN and performance of GMGA in function 8.

number used in tournament operation. In this experiment, GMGA randomly get 3 regions and choose the best. If *PN* is large enough, tournament operation acts as random selection. Thus the results are poor.

Fig. 5 shows how the *PN* changes the performance of GMGA in function 8.

The performance of GMGA keeps improving as the *PN* increase. One reason is that if a region is large enough, the region probably contains plenty of local minima points. A region with plenty of local optima may be regarded as the region with the highest potential to find global optimum. Thus the global optimum can't be obtained. Another reason contributing to this situation is that GMGA have to search in a large space when *PN* is small. It is difficult for GMGA to finds the optimum for lacks of local search ability.

V. CONCLUSION

This paper presents a novel guide mutation to improve the performance of evolutionary algorithms. To validate the proposed guide mutation, we combine it into a SGA and form a guided mutation GA (GMGA).23 benchmark functions with different characteristics are used to investigate the performance of GMGA. The experiment results show that guided mutation operation is very effective in improving the performance of SGA.

REFERENCES

[1] J.H Holland, "Adaption in natural and artificial systems", Ann Arbor, MI: Univ. of Michigan Press, 1975.

[2] W.N. Chen, Jun ZHANG, Henry Chung, W.L., Zhong, W.G. Wu and Y.H., Shi, "A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems", IEEE Transactions on Evolutionary Computation, Vol.14,No.2, pp.278-300, April 2010

[3] Jun ZHANG, Zhi-hui Zhan, Ying Lin, Ni Chen, Yue-jiao Gong, Henry S.H. Chung, Yun Li and Yu-hui Shi, "Evolutionary Computation Meets Machine Learning: A Survey", IEEE Computational Intelligence Magazine, pp.68-75, Nov 2011

[4] Shajulin Benedict, Vasudevan V , "Scheduling of scientific workflows using Niched Pareto GA for Grids," Service Operations and Logistics, and Informatics, 2006. SOLI '06. IEEE International Conference on , vol., no., pp.908-912, 21-23 June 2006

[5] W.N. Chen and Jun ZHANG, "Ant Colony Optimization Approach to Grid Workflow Scheduling Problem with Various QoS Requirements", IEEE Transactions on Systems, Man, and Cybernetics--Part C: Applications and Reviews, Vol. 31, No. 1,pp.29-43,Jan 2009

[6] X. M. Hu, J. Zhang, and et al., "Hybrid Genetic Algorithm Using a Forward Encoding Scheme for Lifetime Maximization of Wireless Sensor Networks," IEEE Transactions on Evolutionary Computation, vol.14, no.5, pp.766-781, Oct. 2010.

[7] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Univ. of Michigan, Ann Arbor, 1975, Diss. Abstr. Int. 36(10), 5140B, University Microfilms no. 76−9381.

[8] J. D. Schaffer, R. A. Caruana, L. J. Eshelman, and R. Das, "A study of control parameters affecting online performance of genetic algorithms for function optimization," in Proc. 3rd Int. Conf. on Genetic Algorithms. San Mateo, CA: Morgan Kaufmann, 1989, pp. 51–60.

[9] T. Back, "Optimal mutation rates in genetic search," in Proc. 5th Int. Conf. on Genetic Algorithms, S. Forrest, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 2–8.

[10] Z.H. Zhan, Jun ZHANG, Y. Li and Henry Chung, "Adaptive Particle Swarm Optimization", IEEE Transactions on Systems, Man, and Cybernetics--Part B. VOL. 39, NO. 6, Dec 2009, Page 1362-1381

[11] T. Back and M. Schutz, "Intelligent mutation rate control in canonical genetic algorithms," in Foundations of Intelligent Systems, 9th Int. Symp., ISMIS'96 (Lecture Notes in Artificial Intelligence, vol. 1079), Z W. Ras and M. Michalewicz, Eds. Berlin, Germany: Springer, 1996, pp. 158–167.

[12] J. Smith and T. C. Fogarty, "Self adaptation of mutation rates in a steady state genetic algorithm," in Proc. 3rd IEEE Conf. on Evolutionary Computation. Piscataway, NJ: IEEE Press, 1996, pp. 318–323.

[13] Jun ZHANG, Henry Chung and W.L., LO, "Clustering-Based Adaptive Crossover and Mutation Probabilities for Genetic Algorithms",IEEE Transactions on Evolutionary Computation Vol.11, No.3, June 2007, Page. 326-335.

[14] Back T,Hoffimeister F and Schwefel H P," A survey of evolution strategies", In Proc of the 4th Int. Genetic Algorithms Conference, CA: Morgan Kaufmann Publishers, 2-9, 1991.

[15] Fogel D B," An introduction to simulated evolutionary optimization", IEEE Trans , Neural Network, vol.5, no.1, 3-14, 1994.

[16] Wei C.J,Yao S.S and He Z.Y., "A modified evolutionary programming", In Proc 1996 IEEE Int. Evolutionary Computation Conference, NJ, IEEE Press, 135-138

[17] Rudolph G, "Local convergence rates of simple evolutionary algorithms with Cauchy mutations", IEEE Trans, Evolutionary Computation, vol.1, no.4, 249-258, 1997.

[18] Gao Ye, Zheng Tao," Improved Genetic Algorithm Based on Chaotic Mutation Operation and Its Application.", In Multimedia Technology (ICMT), 2010 International Conference, pp. 1-3, 2010.

[19] J. H. Zhong, X. M. Hu, J. Zhang and M. Gu, "Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms," in Proc. IEEE Computational Intelligence for Modeling, Control and Automation 2005, vol.2, pp. 1115-1121, Nov. 2005.

[20] Xin Yao, Yong Liu and GuanGIng Lin, "Evolutionary Programming Made Faster", IEEE Trans, Evolutionary Computation, vol.3,NO.2,July 1999, pp 82-102.