Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 185 (2007) 1098–1108
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

Interactive genetic algorithms with multi-population adaptive hierarchy and their application in fashion design

Dun-Wei Gong [a,*], Guo-Sheng Hao [a,b], Yong Zhou [a], Xiao-Yan Sun [a]

[a] School of Information and Electrical Engineering, China University of Mining and Technology, Xuzhou, Jiangsu 221008, PR China
[b] School of Technology, Xuzhou Normal University, Xuzhou, Jiangsu 221011, PR China

Abstract

Limitations of existing interactive genetic algorithms are analyzed and interactive genetic algorithms with multi-population adaptive hierarchy proposed. A model for interactive genetic algorithms with multi-population is established and a strategy for individuals' migration is designed. Adaptive genetic operators are applied to interactive genetic algorithms with a single population, and when a condition for hierarchy is met, the algorithms will evolve in the subspace of the original search space. The algorithms' implementation based on local networks is also presented. The algorithms proposed in this paper can maintain the diversity of populations as a whole, improve abilities in exploitation and exploration, prevent those good individuals from being eliminated and alleviate users' fatigue. The application of the algorithms in fashion design validates that they are feasible and efficient.

Keywords: Genetic algorithms; Multi-population; Adaptation; Hierarchy; Interaction; Fashion design

1. Introduction

Interactive genetic algorithms (IGAs) are simply genetic algorithms (GAs) in which individuals' fitness is based on users' subjective evaluations [1]. Combining users' evaluation abilities with GAs and without any explicit objective function of some optimized problems, IGAs can enlarge the application areas of GAs. Since they were proposed in 1990s, IGAs have been successfully applied in such fields as fashion design, music composition, language handling and rhyme control, knowledge acquisition and data mining [2–5].

Canonical interactive genetic algorithms (CIGAs) adopt a single population and some canonical genetic operators. Individuals' fitness is based on users' subjective evaluations. Compared with tireless computers, users are apt to be tired. Moreover, the number of individuals displayed in the same generation is restricted by human–machine interfaces and the characteristics of system outputs, resulting in a small size of population and a small number of evolutionary generations. In general, the size of population is not more than 10 and the

* Corresponding author.
E-mail address: dwgong@vip.163.com (D.-W. Gong).

0096-3003/$ - see front matter
doi:10.1016/j.amc.2006.07.043

number of evolutionary generations is not more than 20. However, for multi-variable and complicated optimized problems, it is difficult to find optima by using such limited population size and evolutionary generations. Therefore, the optimal results are usually unsatisfactory.

In order to improve the optimization performance of IGAs, there are three problems that should be solved, which are (1) how to alleviate users' fatigue, (2) how to maintain the diversity of a population, hence improving the algorithms' abilities in exploration, and (3) how to improve the algorithms' abilities in exploitation, hence speeding up the algorithms' convergence on condition of finding optima.

The first problem is one of the research foci in IGAs community. For this problem, computers are used to learn the process of users' subjective evaluations and to replace the users in evaluating parts of (or all) the individuals' fitness in an appropriate time so as to reduce the time spent by the users in evaluating individuals and alleviate the users' fatigue, or increase the size of population and the number of evolutionary generations on condition of not changing the degree of the users' fatigue, hence improving the algorithms' performance. Biles et al. adopted neural networks (NNs) to learn users' subjective evaluations and to replace the users in evaluating individuals' fitness [6]. Zhou et al. proposed phase estimations of individuals' fitness based on neural networks in IGAs. In different phases, different NNs are adopted to learn users' subjective evaluations and thereafter the NNs replace the users in evaluating individuals' fitness. The analysis of algorithms and the experiment validated their efficiency [7]. In addition, Hao et al. introduced the concept of gene units and estimated some individuals' fitness by using the estimations of the gene units' fitness [8]. Yamamoto et al. estimated some individuals' fitness by calculating the distances between the estimated individuals and other individuals whose fitness had already been evaluated by users [9].

The second problem is the premise for IGAs to evolve efficiently. Unemi evolved several gene parts independently and prevented those good individuals' gene parts from mutating, thus improving the algorithms' abilities in exploration [10]. The authors think that there are two methods to solve this problem. The first one is to employ IGAs with multi-population, in this method the diversity of populations can be increased by individuals' migration among populations. The second one is to apply mature adaptive genetic operators to IGAs to improve their abilities in exploration. Ref. [10] belongs to the later. In Section 2.1, IGAs with multi-population will be expounded, and in Section 2.2, closed crossover avoidance and adaptive one-point mutation will be expatiated. There are two kinds of implementation for IGAs with multi-population, namely implementation based on a single PC and based on networks. It is obvious that the implementation based on a single PC is not appropriate, because several users cannot perform on the same PC simultaneously. In addition, a single PC not only runs IGAs but also performs individuals' migration among populations, hence making a heavy burden of PC and spending much time. For implementation based on local networks, the algorithms can be allocated to different PCs as clients and the migration of individuals among populations can be implemented on the server. What's more, the evolution of multi-user is helpful for maintaining the diversity of populations as a whole. For implementation based on Internet, the application areas of IGAs can be further expanded, laying a foundation for simultaneous evolutions in different places. In Section 3, implementation of IGAs with multi-population based on local networks will be considered, while that based on Internet will be expatiated in the subsequent paper.

The third problem is the premise for IGAs to find optima. In order to improve the algorithms' local search ability, Lee and Cho directly manipulated the best individual when populations evolved to a certain phase [11]. However, they are just enumerative searches in subspaces and so the IGAs need to be further improved. The authors applied hierarchy to IGAs with a single population and proposed IGAs with hierarchy. The efficiency of IGAs was validated by the practice in fashion design [12]. However, the above results are unsuitable for IGAs with multi-population. In Section 2.3, the issue of hierarchy combining with IGAs with multi-population will be expatiated.

2. IGAs with multi-population adaptive hierarchy

The methodology of interactive genetic algorithms with multi-population adaptive hierarchy (MPAHIGAs) is as follows: (1) A model for IGAs with multi-population is considered and a strategy for migration is designed to maintain the diversity of populations as a whole and to avoid premature convergence, hence improving the algorithms' abilities in global search, (2) adaptive genetic operators are adopted in IGAs with

a single population to improve the efficiency of genetic operators, the abilities of algorithms in exploration and exploitation and to prevent those good individuals from being eliminated, and (3) a strategy for hierarchy is applied to IGAs with multi-population to lead the search in the subspace of the original search space when they evolve to a certain phase in order to alleviate users' fatigue and speed up convergence.

In order to implement the above methodology, three problems need to be solved, namely, which are (1) the model of IGAs with multi-population, (2) some novel adaptive genetic operators in IGAs with a single population, with closed crossover avoidance and adaptive one-point mutation being adopted, and (3) the condition on hierarchy of IGAs with multi-population and determination of the search subspace. They will be expounded in the following sections, respectively.

2.1. Models of IGAs with multi-population

The so-called model of IGAs with multi-population is that the algorithms are composed of several populations and all populations evolve based on IGAs. The number of populations is determined according to optimized problems and their search spaces. All populations initiate and evolve in the whole search space independently. The best individuals of these populations are exchanged among them in every population evolutionary period. All new populations are obtained after replacement based on a certain rule and then evolve sequentially until they find satisfactory individuals.

For convenient illustrations, let $\vec{X}_1(t), \vec{X}_2(t), \ldots, \vec{X}_M(t)$ be M populations in the generation of t in search space Ω, whose population size are $n_1, n_2, \ldots, n_M$, respectively. $\vec{X}_m(0)$ is initiated randomly in Ω, where $m = 1, 2, \ldots, M$. First, the fitness of individuals from different populations is evaluated by different users, and then selection, crossover and mutation operators operate on individuals. Crossover and mutation operators will be given in Section 2.2. Because fitness of individuals from different populations is evaluated by different users, and different users perhaps have different opinions on the same index, fitness of an individual evaluated by different users may not be exactly the same. Let $f_m(x_m(t))$ be fitness of $x_m(t)$ in $\vec{X}_m(t)$. So $f_m(\cdot)$ may not be equal to $f_i(\cdot)$ if $i \neq m$, and $f_m(\cdot)$ cannot be expressed by an explicit function.

Let $x_m^B(t)$ be the best individual in $\vec{X}_m(t)$ when t is the multiple of the population evolutionary period. If the number of the best individuals is more than one, then let $x_m^{B_1}(t), x_m^{B_2}(t), \ldots, x_m^{B_{K_m(t)}}(t)$ be the best individuals, where $K_m(t)$ is the number of the best individuals. Choose $x_m^{B_k}(t), k \in \{1, 2, \ldots, K_m(t)\}$ randomly as the candidate to be exchanged. Choose $j \in \{1, 2, \ldots, M\}, j \neq m$ and let $\vec{X}'_j(t) = \vec{X}_j(t) \cup \{x_m^{B_k}(t)\}$. Hence the size of $\vec{X}'_j(t)$ is $n_j + 1$, and the fitness of $x_m^{B_k}(t)$ in $\vec{X}'_j(t)$ is $f_j(x_m^{B_k}(t))$.

Considering $\vec{X}_j(t)$ and let $x_j^W(t)$ be the worst individual in $\vec{X}_j(t)$. If the number of the worst individuals is more than one, let $x_j^{W_1}(t), x_j^{W_2}(t), \ldots, x_j^{W_{L_j(t)}}(t)$ be the worst individuals, where $L_j(t)$ is the number of the worst individuals. Choose $x_j^{W_l}(t), l \in \{1, 2, \ldots, L_j(t)\}$ randomly, if $f_j(x_j^{W_l}(t)) \leqslant f_j(x_m^{B_k}(t))$, then let $\vec{X}_j(t) = \vec{X}'_j(t) \setminus \{x_j^{W_l}(t)\}$.

2.2. Closed crossover avoidance and adaptive one-point mutation

Closed crossover avoidance and adaptive one-point mutation, are adopted in all populations to improve the efficiency of genetic operators and the abilities in exploration and exploitation of algorithms and to prevent those good individuals from being eliminated.

Without loss of generality, binary strings are adopted to encode decision variables. The methodology of closed crossover avoidance is as follows. If the Hamming distance between two individuals is great, then these two individuals are chosen as candidates to perform crossover, otherwise another individual in the same population is chosen. The lower limit of the Hamming distance changes with the evolutionary phase.

Let T_0 be the evolutionary period of population, and $x_m^i(t)$ and $x_m^j(t)$ be the ith and the jth individuals in $\vec{X}_m(t)$, respectively. If $H(x_m^i(t), x_m^j(t)) > \rho(t) H(\vec{X}_m(t))$, then $x_m^i(t)$ and $x_m^j(t)$ are chosen as candidates to perform crossover, otherwise $x_m^{j_0}(t) \in \vec{X}_m(t)$ satisfying $j_0 = \arg \max_{j \in \{1,2,\ldots,n_m\}} \{H(x_m^i(t), x_m^j(t))\}$ is chosen, which, together with the chosen $x_m^i(t)$ are used as candidates, where $H(x_m^i(t), x_m^j(t))$ represents the Hamming distance between $x_m^i(t)$ and $x_m^j(t)$, $H(\vec{X}_m(t))$ represents the average Hamming distance of $\vec{X}_m(t)$, and $\rho(t)$ is an impact factor reflecting an evolutionary phase, whose mathematical description was given in Ref. [13].

When the two candidates perform crossover, if a crossover point is chosen such that the Hamming distance of the segments being exchanged is zero, then no new offspring will be produced by crossover operator. Hence it is necessary to choose a new crossover point again by the following method.

Let $x_m^i(t)$ and $x_m^j(t)$ be two candidates to perform crossover, c be a crossover point, $\mathrm{Seg}(x_m^{ic1}(t))$ and $\mathrm{Seg}(x_m^{jc1}(t))$ be segments of $x_m^i(t)$ and $x_m^j(t)$ before c, respectively, and $\mathrm{Seg}(x_m^{ic2}(t))$ and $\mathrm{Seg}(x_m^{jc2}(t))$ be segments of $x_m^i(t)$ and $x_m^j(t)$ after c, respectively. If $H(\mathrm{Seg}(x_m^{ic1}(t)), \mathrm{Seg}(x_m^{jc1}(t))) = 0$, then $c \in \{1, 2, \ldots, L\}$ is chosen as a crossover point, whose value is $(\min\{c | c \in \{1, 2, \ldots, L\}, H(\mathrm{Seg}(x_m^{ic2}(t)), \mathrm{Seg}(x_m^{jc2}(t))) > 0\} + \max\{c | c \in \{1, 2, \ldots, L\}, H(\mathrm{Seg}(x_m^{ic2}(t)), \mathrm{Seg}(x_m^{jc2}(t))) > 0\})/2$, where L is the length of individuals' codes.

The methodology of adaptive one-point mutation is as follows. One-point mutation is adopted for the candidate, and its mutation probability is related to the evolutionary ability of population and fitness of the candidate. The weaker the evolutionary ability of population is, the greater its mutation probability will be; and the less the fitness of the candidate is, the greater its mutation probability will be. Let $f_m^B(t) \hat{=} f_m(x_m^B(t))$ be the best fitness of $\vec{X}_m(t)$ in the generation of t. Define:

$$D_m(t) = 1 - \frac{f_m^B(t-1)}{f_m^B(t)} \tag{1}$$

as $\vec{X}_m(t)$'s evolutionary ability, then $D_m(t) \in (0, 1)$.

Let $f_m^i(t) \hat{=} f_m(x_m^i(t))$ be $x_m^i(t)$'s fitness in $\vec{X}_m(t)$. Then $x_m^i(t)$'s mutation probability is set according to the following formula:

$$p(x_m^i(t)) = \mu \mathrm{e}^{-D_m(t)} + (1-\mu)\left(1 - \frac{f_m^i(t)}{f_m^B(t)}\right), \tag{2}$$

where $\mu \in (0, 1)$ is a factor to balance $\vec{X}_m(t)$'s evolutionary ability and $x_m^i(t)$'s fitness, which can also be chosen according to users' preference.

2.3. Condition on hierarchy and search subspace of IGAs with multi-population

The problem on hierarchy combining with IGAs with a single population was considered in Ref. [12] and the condition on hierarchy is as follows. When the deviation of the individuals' fitness is less than a given value, the algorithms will evolve in a certain subspace of the original search space. However, the above condition is unsuitable for IGAs with multi-population, because different users may give different evaluations for the same individual. Hence the best individual in $\vec{X}_i(t)$ may be thought a bad one by the user of $\vec{X}_j(t)$.

For the reason mentioned above, the following methods to determine the condition on hierarchy and search subspace are proposed. Consider the best individual in a population and the immigrant individuals from other populations. If the maximal deviation of the evaluations for these two kinds of individuals from the same user is smaller than a given value, all the users involved in the evolutions have almost the same opinion on the best individual. So the best individual of IGAs with multi-population in global search phase can be determined. The subspace of the original search space is determined based on the best individual's codes. The algorithms evolve in the above subspace to improve their abilities in exploitation. The algorithms continue these steps until satisfactory individuals are found. For convenient, only two layers are considered here, namely, the global search and the local search in the preferred subspace.

Consider $\vec{X}_m(t)$, the best individual before migration is $x_m^B(t)$, and the immigrant individual from $\vec{X}_j(t)$ is $x_j^B(t)$. The user of $\vec{X}_m(t)$ evaluates their fitness as $f_m(x_m^B(t))$ and $f_m(x_j^B(t))$, respectively. Let

$$\delta(t) = \max_{\substack{m \in \{1,2,\ldots,M\} \\ j \in \{1,2,\ldots,M\}}} |f_m(x_m^B(t)) - f_m(x_j^B(t))|. \tag{3}$$

If $\delta(t_{\mathrm{gs}}) \leqslant \Delta$, the algorithms will begin to evolve in the local search space, where Δ is a given threshold, t_{gs} is the terminated generation in the global search phase, and $x^{gB}(t_{\mathrm{gs}})$ corresponding to $\max_{\substack{m \in \{1,2,\ldots,M\} \\ j \in \{1,2,\ldots,M\}}} \{f_m(x_m^B(t)), f_m(x_j^B(t))\}$ is the best individual in the global search phase.

Let the best individual's codes in the global search phase be

$$x^{gB}(t_{\mathrm{gs}}) = x^{gB_1} x^{gB_2} \ldots x^{gB_{SL}}, \tag{4}$$

where x^{gB_i} is the segment codes corresponding to the ith part of the best individual's phenotypes and its length is l_i, $i = 1, 2, \ldots, SL$, where SL is the number of the segments. If all the users think that the jth part of the best individual's phenotypes needs improving, and without loss of generality, it is assumed that only one part needs improving, then $\Omega_l = \{x^{gB_1} x^{gB_2} \ldots x^{gB_{j-1}} *^{l_j} x^{gB_{j+1}} \ldots x^{gB_{SL}}\}$ is the subspace in the local search phase, where $*^{l_j}$ is a string composed of 0 and 1, and its length is l_j. It can be seen from the expression of Ω_l that the subspace in the local search phase is obviously smaller than that in the global search phase, which can speed up the algorithms' convergence and improve abilities in exploitation.

After the subspace in the local search phase has been determined, new M populations with size of n_m are initiated in the subspace under some rules and go on evolving according to Sections 2.1 and 2.2, until the best individual which satisfies all users involved is found.

Because the algorithms evolve in the subspace during the local search phase, it is clear that for populations with the same size, the sampling probability of the point in the subspace is greater than that in the whole space. Hence the probability of finding the optimal solution in the subspace is great. Moreover, because only parts of the individuals' phenotypes are evaluated, users are quite easy to distinguish different individuals, which alleviates their fatigue and improve the search precision of the algorithms. Hence it is easy to find the optimal solution in the subspace.

2.4. Steps of algorithms

The steps of MPAHIGAs proposed in this paper are as follows:

Step 1: Determine the control parameters of algorithms.
Step 2: Let $t = 0$ and initiate M populations $\vec{X}_1(t), \vec{X}_2(t), \ldots, \vec{X}_M(t)$ according to certain rules.
Step 3: For $\vec{X}_m(t), m = 1, 2, \ldots, M$, a user evaluates its individuals' fitness and it evolves for a population evolutionary period by proportional selection, crossover and mutation operators in Section 2.2.
Step 4: Choose the best individuals of all populations, and then migrate them according to Section 2.1.
Step 5: Judge whether it is the time for the algorithms to terminate the global search phase according to formula (3). If yes, then go to step 6, otherwise go to step 3.
Step 6: Choose the best individual $x^{gB}(t_{gs})$ in the global search phase according to formula (4) and determine the subspace according to Section 2.3.
Step 7: Initiate M populations, namely, $\vec{X}_1(t), \vec{X}_2(t), \ldots, \vec{X}_M(t)$ in the subspace according to certain rules.
Step 8: For $\vec{X}_m(t), m = 1, 2, \ldots, M$, a user evaluates its individuals' fitness and it evolves for a population evolutionary period by proportional selection, crossover and mutation operators in Section 2.2.
Step 9: Choose the best individuals of all populations and judge whether the satisfactory solution is found. If yes, then go to step 10, otherwise migrate individuals according to Section 2.1 and go to step 8.
Step 10: Output the satisfactory individual and end the algorithms.

2.5. Effect of the number of populations on algorithms' performance

The number of populations affects the algorithms' performance to some degree. Hence it is important to determine appropriate population numbers. In general, the number of populations is determined by optimized problems and it is different for different problems. On the one hand, the larger the number of populations, the better the whole populations' diversity is, hence the larger the possibility of the algorithms to find better solutions will be. On the other hand, the larger the number of populations, the slower the speed of hierarchy will be, which can be seen from Section 2.3. Besides, the larger the number of populations, the more the resources for calculation of the algorithms is needed. Hence for complicated optimization problems, when one wants to find good solutions, the number of populations is often large. But for some simple optimization problems, it is sufficient for a small number of populations.

3. Implementation of algorithms

For the limitations of implementing IGAs with multi-population based on a single PC, MPAHIGAs proposed in this paper are implemented based on local networks. All clients are used to store private data of IGAs

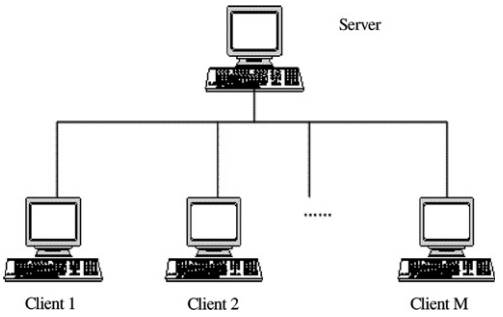

Fig. 1. Topological structure of algorithms.

with a single population, to implement IGAs with a single population, to save the best individuals in current population, and to access the best individuals from other populations. The running platform of all clients is the same and only IGAs with a single population evolve in each of all clients, indicating that M clients are needed. The server is to manage all clients, to store all immigrant individuals, and to implement all individuals' migration. What runs on all clients are IGAs platform for clients, and what runs on the server is SQL Server 2000. The topological structure of the algorithms is shown in Fig. 1.

Each client saves its best individual to the server for every population evolutionary period. After all individuals exchanged in the server according to Section 2.1, the clients access their immigrant individuals.

All clients perform IGAs with a single population. Because the fitness of individuals is evaluated by users and users are easy to be tired, the population size and the evolutionary generations should not be too large. In general, the population size and the evolutionary generations should not be larger than 10 and 20, respectively, and the evolutionary period should be 4 or 5. However, for different users some fluctuation is permitted within the above ranges. Further, other parameters such as a balance factor can also vary within a certain range so as to reflect users' preference.

In order to alleviate users' fatigue as possible as one can, it is important to design good human machine interfaces. In general, they should be friendly and simple.

4. Experiments and result analysis

MPHAIGAs are applied to fashion design to validate their feasibility and efficiency.

4.1. Experiment settings

The object of experiment is fashion design. The aim of fashion design is to obtain "good design". However, different users have different opinions on "good design". Hence it is impossible to obtain a general fitness function and canonical GAs are unsuitable for fashion design. Because individuals' fitness in IGA is directly determined by users, IGAs are suitable for fashion design. Kim and Cho expounded the feasibility of the application of IGAs in fashion design and did some experiments [2].

In general, women fashions are divided into three parts: namely, sleeve, neckline and skirt. In addition, each part is encoded with a 4-bit binary string in which the first 2 bits are for style and the rest 2 bits for color. Therefore, there are four kinds of styles and colors for each part. The size of the search space is $2^{12} = 4096$. Fig. 2 shows the interface for setting experiment parameters in clients. The common parts of all clients are as follows. The population size is 8, the population evolutionary period 4, the threshold value $\Delta = 10$, the time to terminate the algorithms is determined by users. Here the elitist strategy is adopted.

There are two kinds of candidates to mutate. One is the individual whose fitness is known, the other is the individual generated by crossover operator whose fitness is unknown. For the latter one, users should evaluate its fitness. As the probability of mutation is very small and the mutation operator only happens to individuals with low fitness, it has little influence on users' fatigue. In addition, mutating individuals with low fitness is helpful for speeding up the algorithms' convergence, hence decreasing the evolutionary generations.

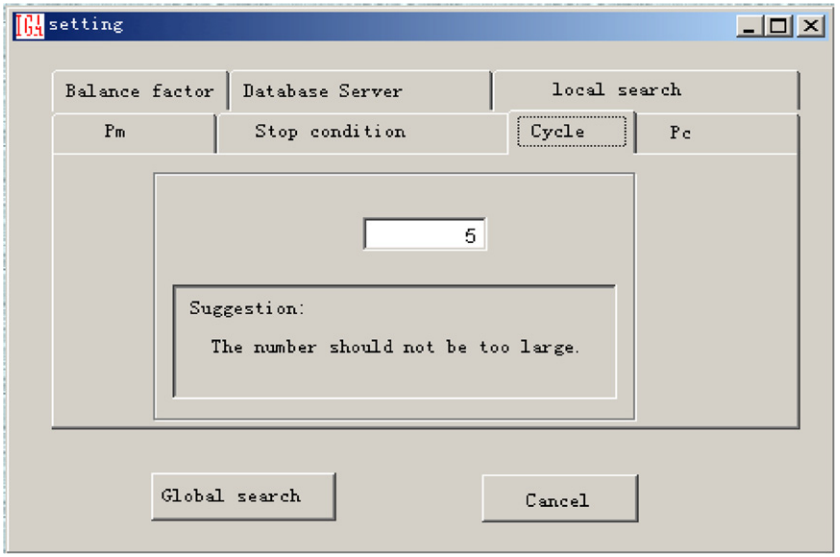

Fig. 2. Interfaces of experiment settings in all clients.

The parameters that can be different in different clients include crossover probability and balance factor. If the above parameters are out of their ranges, the system will inform users to set them again.

4.2. Experiment processes

After experiment settings, users begin to do experiment. The number of populations is 6. All clients will call system command, namely ipconfig, to obtain automatically their MACs (machine address of network cards, the unique ID numbers in the server) by which the server recognizes different best individuals from different clients. When the algorithms evolve to multiples of the population evolutionary period, users firstly click on "migrate" button to migrate their best individuals, then codes of the best individuals and MACs of the clients are submitted to the database server. After that, users click on "immigrate" button to immigrate other clients' best individuals and display them. Fig. 3 shows the interface of individuals' immigration.

In Fig. 3 the first individual is the best individual from its own population and the other five individuals are from other populations. Then users reevaluate all immigrant individuals' fitness. When they click on "OK" button, the individual with the highest fitness will replace the individual with the lowest fitness. When

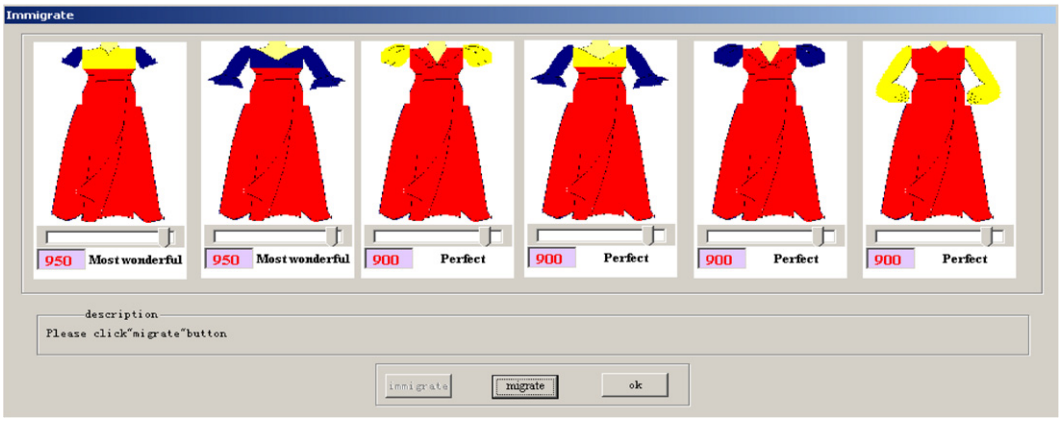

Fig. 3. Interfaces of individuals' immigration.

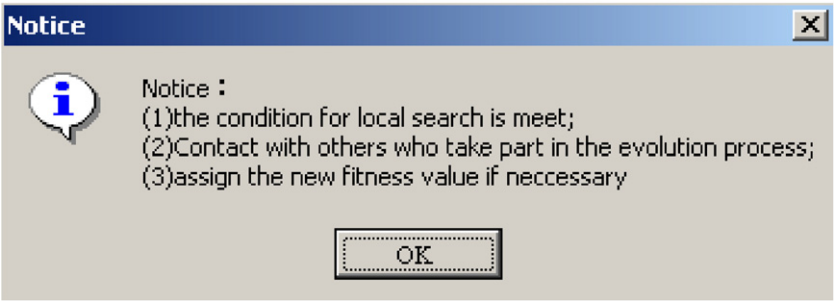

Fig. 4. Information given by system when algorithms begin to local search.

$\delta(t_{gs}) \leqslant \Delta$ is met in the global search phase, the algorithms will begin the local search. The information given by the system is shown in Fig. 4.

In addition, before the local search, all users will negotiate with each other to determine the best individual that is the base for initiating populations in the local search phase. Fig. 5 shows that all users choose the best individual whose sleeve and skirt are kept and only neckline needs a further evolution. In addition, the interface like Fig. 2 will appear to let all users set parameters again.

In IGAs, friendly interface can alleviate users' fatigue. During the experiment, the default individuals' fitness is 500, and based on this value, the users change accordingly. This is the first method to alleviate users' fatigue. The second friendly method is to display the best individual in the last generation to users (Fig. 6). Its advantage is that users can find the difference in phenotypes between the best individual in the current generation and that in the last generation, which is helpful for users to understand the evolutionary process. The third friendly method is to adopt fuzzy languages to describe users' preference. For example, in Fig. 6 the fourth individual's fitness is 900 and whose fuzzy language is "the most beautiful". The fourth friendly method is to apply slider bars for users to evaluate individuals' fitness. What users have to do in evaluation is only to pull slider bars. Surely, if users like, they can also input individuals' fitness in textboxes. The last friendly method is to adopt discrete fitness [14]. There are 20 scales between 0 and 1000, in which users can choose one according to their requirements.

4.3. Comparative experiments and results

In order to validate MPHAIGAs' efficiency, the following comparative experiments are performed, namely, IGAs with multi-population vs IGAs with a single population, closed crossover avoidance vs one-point

Fig. 5. Interfaces before local search.

Fig. 6. Main interfaces of evolutions.

crossover, adaptive one-point mutation vs one-point mutation, and hierarchy vs no hierarchy. These comparative experiments and results are expounded as follows:

(1) *IGAs with multi-population vs IGAs with a single population*

For all algorithms, 20 times of independent runs are performed. The average of standard deviations of individuals' fitness in the fifth and the ninth generation is shown in Table 1 and the average of the best individuals' fitness in the fifth and the ninth generation is shown in Table 2.

The average convergence generation is 8.4 for IGAs with multi-population and 17.2 for IGAs with a single population. It can be seen from Tables 1 and 2 that IGAs with multi-population can maintain the diversity of populations, improve the best individual's quality and speed up algorithms' convergence.

(2) *Closed crossover avoidance vs one-point crossover*

One evolution of the algorithms is chosen and new populations in some generation obtained from closed crossover avoidance and one-point crossover respectively are considered. The average Hamming distance of the new populations is shown in Table 3. It can be seen from Table 3 that closed crossover avoidance can make populations have better performance in maintaining the diversity of populations than one-point mutation.

Table 1
Average of standard deviations of individuals' fitness

No. of generations	Average of standard deviations of individuals' fitness	
	IGAs with a single population	IGAs with multi-population
5	70.7	295.1
9	35.3	313.3

Table 2
Average of the best individuals' fitness

No. of generations	Average of the best individuals' fitness	
	IGAs with a single population	IGAs with multi-population
5	755	795
9	785	835

Table 3
Average Hamming distance of new populations

	Average Hamming distance of new populations
Closed crossover avoidance	2.3
One-point crossover	1.8

Table 4
Results of adaptive one-point mutation vs one-point mutation

	Average Hamming distance	Correlation coefficients between individuals' fitness and their mutation probabilities	Correlation coefficients between population evolutionary abilities and individuals' mutation probabilities
Adaptive one-point mutation	2.4	−0.47	−0.10
One-point mutation	1.6	0.02	0

(3) *Adaptive one-point mutation vs one-point mutation*

One evolution of the algorithms is chosen and new populations in some generation obtained from adaptive one-point mutation and one-point mutation are respectively considered. The average Hamming distance of new populations, correlation coefficients between individuals' fitness and their mutation probabilities, and correlation coefficients between population evolutionary abilities and individuals' mutation probabilities are shown in Table 4. It can be seen from Table 4 that adaptive one-point mutation can prevent those good individuals from being eliminated.

(4) *Hierarchy vs no hierarchy*

During the experiment, all users think that skirt and sleeve parts are satisfactory and neckline needs a further evolution. The ratio of the search space after and before hierarchy is $2^4/2^{12} = 1/2^8 = 0.39\%$. Hence it is evident that the hierarchy can reduce the search space.

5. Conclusion

MPHAIGAs are proposed in this paper after analyzing existing IGAs' limitations. The innovation and advantages of the algorithms are mainly as follows: (1) The algorithms can maintain the diversity of populations as a whole, avoid premature convergence and improve their global search ability by establishing a model for IGAs with multi-population and designing an appropriate strategy for individuals' migration. (2) The algorithms can improve genetic operators' efficiency, their abilities in exploitation and exploration, and prevent those good individuals from being eliminated by applying adaptive genetic operators in all IGAs with a single population. (3) When the hierarchy condition is met, the algorithms will evolve in the subspace of the original search space, which can alleviate users' fatigue and speed up convergence. (4) The algorithms' ability in parallel computation is improved and their application areas are enlarged by implementation based on local networks. The algorithms' application in fashion design indicates that the IGAs are feasible and efficient.

The future research problems include (1) to look for new methods to deal with evaluation conflicts coming from different users, and (2) to implement MPHAIGAs based on Internet and their performance analysis.

Acknowledgments

Project *Research on Key Issues of Co-interactive Evolutionary Computation and their Implementation based on Internet* (60304016) supported by National Natural Science Foundation of China.

References

[1] H. Takagi, Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation, Proceedings of the IEEE 89 (9) (2001) 1275–1296.

[2] H. Kim, S. Cho, Application of interactive genetic algorithm to fashion design, Engineering Applications of Artificial Intelligence 13 (6) (2000) 635–644.
[3] N. Tokui, H. Iba, Music composition with interactive evolutionary computation, in: Proceedings of the 3rd International Conference on Generative Art, 2000, pp. 215–226.
[4] T. Morita, H. Iba, Generating emotional voice and behavior expression by interactive evolutionary computation, in: Proceedings of the 62nd Annual Meeting of Japan Society for Information Processing, 2001, pp. 45–46.
[5] T. Iwasaki, A. Kimura, Y. Todoroki, et al., Interactive virtual aquarium, in: Proceedings of the 5th Annual Conference of the Virtual Reality Society of Japan, 2000, pp. 141–144.
[6] J. Biles, J. Anderson, L. Loggi, Neural network fitness functions for a musical IGA, in: International ICSC Symposium on Intelligent Industrial Automation and Soft Computing, 1996, pp. 39–44.
[7] Y. Zhou, D. Gong, Phase estimations of individuals' fitness based on neural networks in interactive genetic algorithms, Control and Decision 20 (2) (2005) 234–236.
[8] G. Hao, D. Gong, Y. Shi, Search space partition based autonomous genetic algorithms and theirs application, in: Proceedings of the 17th Symposium on China Automation Institute EastChina Region Automation, 2004, pp. 75–78.
[9] M. Yamamoto, T. Hashiyama, S. Okuma, Reducing computational time on evolution under the real environment using fitness estimation, in: Proceedings of the 26th Annual Conference of the IEEE on Industrial Electronics Society, 2000, pp. 2497–2500.
[10] T. Unemi, Partial breeding—a method of IGA for well-structured large scale target domains, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2002.
[11] H. Lee, S. Cho, Analysis of direct manipulation in interactive evolutionary computation on fitness landscape, in: Proceedings of the 2002 Congress on Evolutionary Computation, 2002, pp. 460–465.
[12] D. Gong, G. Hao, Y. Zhou, et al., Interactive evolutionary computations with hierarchy and their application in fashion design, Control and Decision 19 (10) (2004) 1117–1120.
[13] D. Gong, F. Pan, Theory and Applications of Adaptive Genetic Algorithms, Press of China University of Mining and Technology, Xuzhou, 2003, pp. 62–65.
[14] M. Ohsaki, H. Takagi, Improvement of presenting interface by predicting the evaluation order to reduce the burden of human interactive EC operators, in: Proceedings of the IEEE International Conference on System, Man, Cybernetics, 1998, pp. 1284–1289.