An Efficient Genetic Algorithm with Less Fitness Evaluation by Clustering

Hee-Su Kim
MW Lab., Mindware, Co., Ltd.
10th Floor, Wonchang Building, 26-3
Youido-Dong, Youngdeungpo-Ku, Seoul 150-010, Korea
madoka@candy.yonsei.ac.kr

Sung-Bae Cho
Department of Computer Science, Yonsei University
134 Shinchon-dong, Sudaemoon-ku
Seoul 120-749, Korea
E-mail:{madoka,sbcho}@candy.yonsei.ac.kr

Abstract - To solve a general problem with genetic algorithm, it is desirable to maintain the population size as large as possible. In some cases, however, the cost to evaluate each individual is relatively high, and it is difficult to maintain large population. To solve this problem we propose a hybrid GA based on clustering, which considerably reduces evaluation number without any loss of its performance. The algorithm divides the whole population into several clusters, and evaluates only one representative for each cluster. The fitness values of other individuals are estimated from the representative fitness values indirectly, which can maintain large population with less number of evaluations. Several benchmark tests have been conducted and the results show that the proposed GA is very efficient.

1 Introduction

Evolutionary computation (EC) is an efficient method for machine learning, optimization and classification, based on evolution mechanisms such as biological genetics and natural selection. EC provides efficiency and advantages from set of points called population, and improves the population by generations to solve a problem [Chamber95, Goldberg89]. Due to this fact there has been extensive research on EC, making it a major stream of artificial intelligence. It is required to make the population size of evolution as large as possible because EC approach evolves the population spread over the search space. However, in some specific problem the cost to evaluate individuals is relatively high, and this makes it difficult to maintain large number of individuals in a population. Smaller population causes several negative results such as genetic drift.

One example that requires smaller population is interactive evolutionary computation (IEC) application. IEC is a technique that performs optimization based on human evaluation [Takagi98]. A human operator can obtain what he wants through repeated interaction with computer. It has a special advantage, which is to adopt user's choice as fitness, when fitness function cannot be explicitly defined. This property allows IEC application to be applied on artistic domains such as music or design, which are almost impossible to be solved with simple EC. However this kind of approach requires direct evaluation of user for each individual and the fact limits the population size.

To solve this evaluation cost problem, we propose a hybrid GA based on clustering, which considerably reduces the number of evaluation without any loss of its performance. Section 2 introduces GA and clustering algorithm as background. Section 3 describes the hybrid clustering GA we have proposed. Section 4 provides some experimental results and analysis of comparing proposed GA with simple GA.

2 Background

2.1 Genetic Algorithm

GA was proposed by John Holland in early 1970s. It applies some of natural evolution mechanisms such as crossover, mutation, and survival of the fittest to optimization and machine learning. GA provides very efficient search method working on population, and has been applied to many problems of optimization and classification [Goldberg89]. General GA process is as follows [Eberhart96]:

(1) Initialize the population of genes.
(2) Calculate the fitness for each individual in the population.
(3) Reproduce the individuals selected to form a new population according to each individual's fitness.
(4) Perform crossover and mutation on the population.
(5) Repeat step (2) through (4) until some condition is satisfied.

Crossover operation swaps some part of genetic bit string within parents. It emulates just as crossover of genes in real world that descendants are inherited characteristics from both parents. Mutation operation inverts some bits from whole bit string at very low rate. In real world we can see that some mutants come out rarely. Fig. 1 shows the way of applying crossover and mutation operations to genetic algorithm. Each individual in the population evolves to getting higher fitness generation by generation.

As its basic form, GA has several limitations and many developments to solve them are in progress these days. To avoid genetic drift that leads the evolution to local optima, it is suggested to maintain diversity within individuals by several way such as migration model, local selection, and minimal generation gap [Unemi98]. To accelerate the convergence, Seront et al. proposed hybrid GA with local search [Seront00] and Ingu et al. suggested using search space approximation [Ingu99]. To apply GA to artistic domains such as music and design, IGA was proposed [Takagi98].

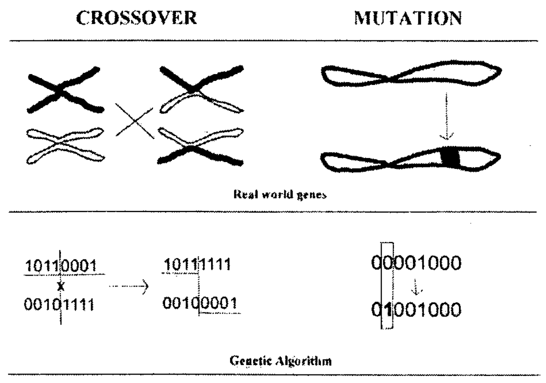

Fig. 1 Crossover and mutation

2.2 Clustering Algorithm

Clustering algorithm refers to the process of grouping samples so that the samples are similar within group [Gose96]. These groups are called clusters. In applying clustering algorithm, it is very important to decide the similarity measure. Most common methods are using distance measures such as city block distance, Euclidean distance, and Minkowski distance [Kandel82]. These methods compute the distance from the notation :

$$d_{ij} = d(X_i, X_j) = \sqrt[m]{\sum_{k=1}^{p} |x_{ik} - x_{jk}|^m} \qquad (1)$$

- $m = 1$: City block distance
- $m = 2$: Euclidean distance
- $m \geq 3$: Minkowski distance

There are three general categories of clustering techniques: Hierarchical clustering, partitional clustering, and overlapping clustering.

A. Hierarchical Clustering

Hierarchical clustering algorithm constructs a structure of clusters. In this structure a cluster can have several sub-structures which are composed of other clusters. Hierarchical clustering algorithm takes two different approaches: agglomerative algorithm of a bottom-up approach and divisive algorithm of a top-down approach. Agglomerative algorithm starts with n clusters, consisting of one sample, and continues to merge most similar clusters. On the other hand, divisive algorithm starts with one huge cluster consisting of all individuals and continues to divide them [Gose96, Haritigan75].

There are several hierarchical clustering algorithms such as single-linkage algorithm, complete-linage algorithm, average-linkage algorithm, and Ward's method, which is also called as minimum-variance method. Ward's method is described in Fig. 2.

```
(1) Assign all samples as clusters with one element.
(2) If a cluster contains $m$ samples $x_1, \ldots, x_m$ where $x_i$ is
    the feature vector $(x_{i1}, \cdots, x_{id})$, compute the squared
    Euclidean distance from the mean
        $\sum_{j=1}^{d} (x_{ij} - \mu_j)^2$
    where
        $\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_{ij}$
(3) Find cluster pair, which has smallest squared error
        $E = \sum_{i=1}^{m} \sum_{j=1}^{d} (x_{ij} - u_j)^2 = m\sigma^2$
    and merge them.
(4) Go to step (2) until all clusters are united.
```

Fig. 2 Ward's algorithm

B. Partitional Clustering

Different from hierarchical clustering which creates a series of nested clusters, partitional clustering usually creates one set of clusters that partition the data into similar groups. Samples close to one another are assumed to be similar and the goal of the partitional clustering algorithms is to group data that are close together. In many of the partitional algorithms, the number of clusters to be constructed is specified in advance. Hard c-means (HCM) algorithm, k-means algorithm, Forgy's algorithm, and isodata algorithm is good examples of partitional clustering [Gose96, Haritigan75]. Fig. 3 describes a simple k-means algorithm.

```
(1) Form $k$ clusters with first $k$ samples
(2) For each of the remaining $n$-$k$ samples
    • Put the sample into the cluster identified with nearest
      centroid
    • Recompute the centroid of altered cluster
(3) For each of all $n$ samples
```

- Put the sample into the cluster identified with nearest centroid

Fig. 3 K-means algorithm

C. Overlapping Clustering

Overlapping clustering algorithm has no hierarchical structure between clusters, similar to partitional clustering. However, this approach does not define closed clusters. In overlapping clustering, each cluster can be overlapped partially with others. Fuzzy c-means (FCM) algorithm and b-clump algorithm are in this category [Höppner99, Xie91]. Fig. 4 describes the FCM algorithm.

For a p by p positive definite matrix, p is the dimension of the vectors $X_j(j = 1, 2, \cdots, n)$, c is the number of clusters, n is the number of vectors (or data points), and $m>1$ is the fuzziness index. To minimize the objective function J_m

$$J_m = \sum_{k=1}^{N}\sum_{i=1}^{c}(\mu_{ij})^m \|X_j - V_i\|^2$$

(1) Initialize memberships μ_{ij} of X_j belonging to cluster i such that

$$\sum_{i=1}^{c}\mu_{ij} = 1$$

(2) Compute the fuzzy centroid Vi for $i = 1, 2, \cdots, c$ using

$$V_i = \frac{\sum_{j=1}^{n}(\mu_{ij})^m X_j}{\sum_{j=1}^{n}(\mu_{ij})^m}$$

(3) Update the fuzzy membership μ_{ij} using

$$\mu_{ij} = \frac{\left(\frac{1}{d^2(X_j,V_i)}\right)^{\frac{1}{(m-1)}}}{\sum_{i=1}^{c}\left(\frac{1}{d^2(X_j,V_i)}\right)^{\frac{1}{(m-1)}}}$$

(4) Repeat steps (2) and (3) until the value of the objective function J_m is no longer decreasing.

Fig. 4 Fuzzy c-means algorithm

Clustering techniques allow the division into subgroups to be done automatically, without any preconception about what kinds of groupings should be found. Cluster analysis has been applied in many fields. In image analysis especially, clustering can be used to find the groups of pixels with similar gray levels, colors, or local texture, in order to discover the various regions in the image [Anderberg73, Fukunaka90].

3 Hybrid GA with Clustering

One possible problem of GA application is genetic drift. It means that the searching is stuck on local optima without any progress to optimal solution. This is because GA optimizes sampled group of individual named population out of whole search space. It is desirable to maintain the population size as large as possible to avoid such a problem.

However, several problems such as interactive genetic algorithm (IGA) application require relatively high cost to evaluate individuals and this makes it difficult to maintain large population. Therefore, to reduce evaluation number efficiently without any loss of performance, we are proposing a hybrid GA with clustering in this section.

The basic idea is to perform the evaluation by two step mechanism. We separate all individuals in the population into subgroups by clustering method, and evaluate one representative of each subgroup. The fitness values of remaining individuals are estimated from this representative fitness of each subgroup. The other GA operation is same as in simple GA. Fig. 5 describes the algorithm in pseudo-code and Fig. 6 shows the basic idea in diagram.

```
Procedure Cluster_GA() {
    Initialize();
    While not end condition do
        SimpleGA();
        Clustering();
        Evaluation();
    End while
}
Procedure SimpleGA() {
    Select();
    Crossover();
    Mutate();
}
Procedure Clustering() {
    /* Performs clustering which divides
       population into k cluster */
}
Procedure Evaluation() {
    Pick_Representatives();
    Evaluate_Representatives();
    Indirect_Evaluation();
}
Procedure Indirect_Evaluation() {
    /* Evaluates non-representative
       individuals indirectly from the
       fitness values of representatives */
}
```

Fig. 5 Proposed GA description

To implement the algorithm, we have to decide the clustering method and indirect evaluation method. For the clustering method we have chosen the k-means algorithm, one of partitional clustering algorithm. The algorithm is one of the simplest partitional algorithms and it is advantageous that we can specify the number of clusters in advance.

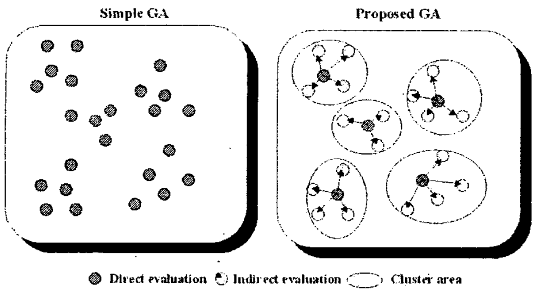

Fig. 6 Simple GA vs. proposed GA

For an indirect evaluation method, there can be various ways to compute fitness value of individuals from representative fitness. We have selected one of the simplest and intuitive methods: the distance from the representative. As described above, the representative is evaluated by objective function and therefore we already know the fitness value of each representative. Remaining individuals get fitness value computed from it, in proportion to a distance from representative. We have used the Euclidean distance and it can be calculated from (1) with m=2. We can get equation

$$dist(x) = \sqrt{\sum_{i=1}^{n}(c_i - x_i)^2}$$

as a result and use it as a distance measure.

4 Experimental Results

To prove that the proposed GA efficiently reduces evaluation number without any loss of the performance, we have conducted several benchmark tests. Eight benchmark functions have been used to compare three genetic algorithms including the proposed GA. We have named the candidate algorithms pop100, clu10, and pop10. Pop100 is simple genetic algorithm that is using 100 individuals as population. Clu10 is the proposed GA, which is using 100 individuals with 10 clusters. In other words, it evaluates only 10 representatives and remaining 90 individuals are evaluated indirectly from the representative fitness values. Pop10 is simple genetic algorithm with 10 population size. We want to prove that the performance of clu10 is as good as one of pop100, though proposed algorithm clu10 evaluates only 10 individuals directly. Table 1 describes on these three candidate algorithms, and Table 2 shows general parameters of GA.

Benchmark functions used for the experiment, the 3D landscapes of them, and the results are as follows. For every benchmark function, clu10 shows almost same performance as pop100 while pop10 shows very poor performance.

Table 1. Candidate GAs used in the experiment.

pop100	Simple GA using 100 individuals
clu10	Proposed GA using *k*-means algorithm for clustering and fitness assigning by distance. The population size is 100 and it evaluates 10 times only, for each cluster's representative.
pop10	Simple GA using 10 individuals

Table 2. General parameters of GA

	pop100	clu10	pop10
Population size	100	100	10
# of evaluation	10	10	10
# of clusters	-	10	-
Crossover rate	0.9		
Mutation rate	0.001		
# of generation	200		

<Function 1> De Jong Function 1

$$f(x) = \sum_{i=1}^{n} x_i^2 \text{ , n=3, } -5.12 \le x_i \le 5.12$$

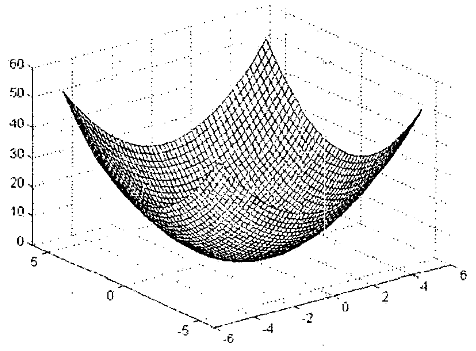

Fig. 7(a) 3D landscape of De Jong function 1

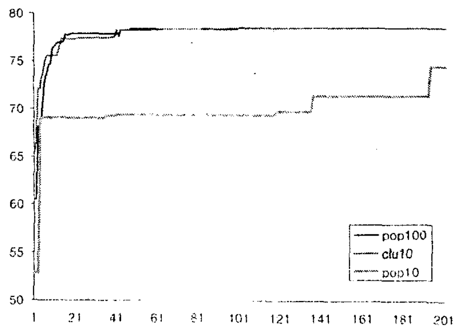

Fig. 7(b) Benchmark result for De Jong function 1

<Function 2> Griewangk Function

$$f(x) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right),$$

$$n=10,\ -600.0 \le x_i \le 600.0$$

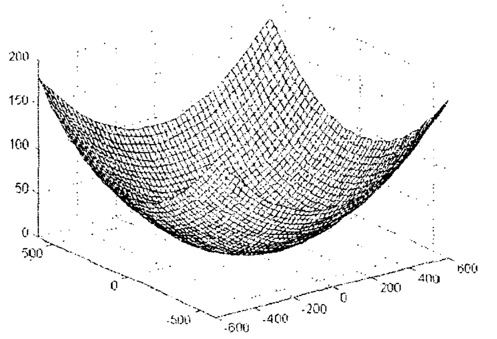

Fig. 8(a) 3D landscape of Griewangk function

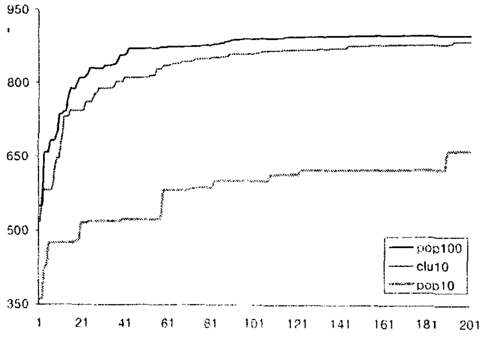

Fig. 8(b) Benchmark result for Griewangk function

Both De Jong function 1 and Griewangk function are typical quadratic function. The difference between them is that De Jong function 1 is complete quadratic function while Griewangk function has product term in it. Therefore the function shows implicit relations between individuals. This property leads optimization algorithm that uses few points to wrong way. As the result, the difference of performance between pop10 and clu10 or pop100 in Fig. 8(b) is much bigger than that of Fig. 7(b).

<Function 3> De Jong Function 2

$$f(x) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2,\ -2.048 \le x_i \le 2.048$$

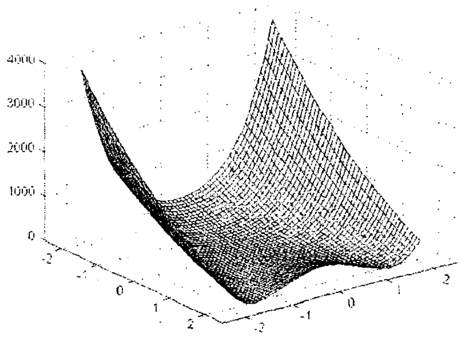

Fig. 9(a) 3D landscape of De Jong function 2

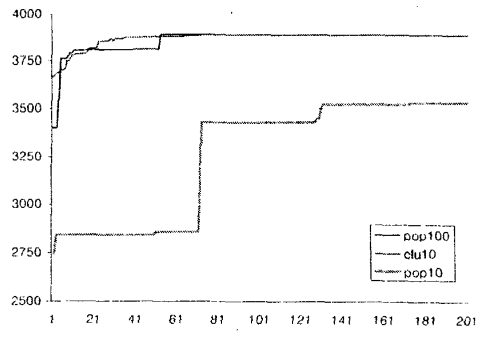

Fig. 9(b) Benchmark result for De Jong function 2

De Jong function 2 has a convex in the center of the landscape. In the evolution process both proposed clu10 and pop100 avoid local optima easily and rapidly, while pop10 is trapped on them for long time.

<Function 4> De Jong Function 3

$$f(x)=\sum_{i=1}^{n}\mathrm{int}(x_i)\ ,\ n=5,\ -5.12\le x_i\le 5.12$$

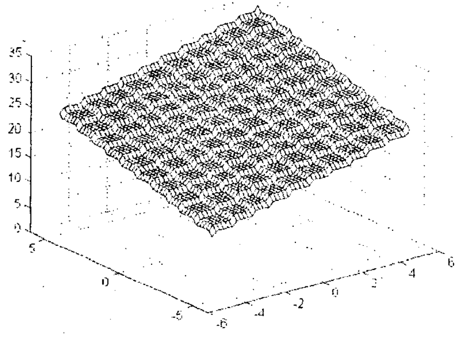

Fig. 10(a) 3D landscape of De Jong function 3

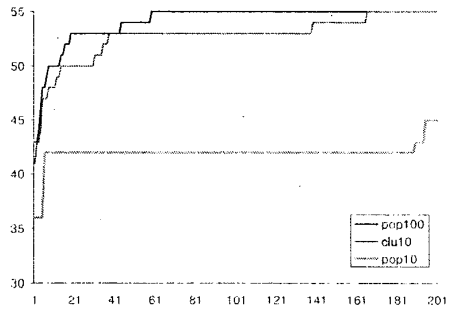

Fig. 10(b) Benchmark result for De Jong function 3

<Function 5> De Jong Function 4

$$f(x)=\sum_{i=1}^{n}(ix_i^4+Gauss(0,1))\ ,\ n=30,\ -1.28\le x_i\le 1.28$$

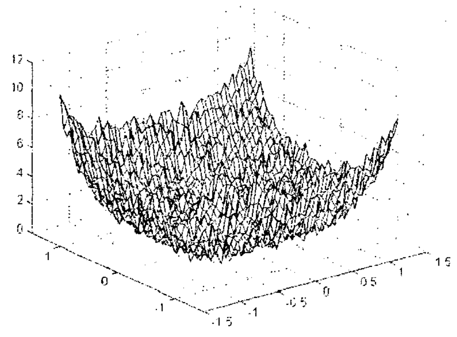

Fig. 11(a) 3D landscape of De Jong function 5

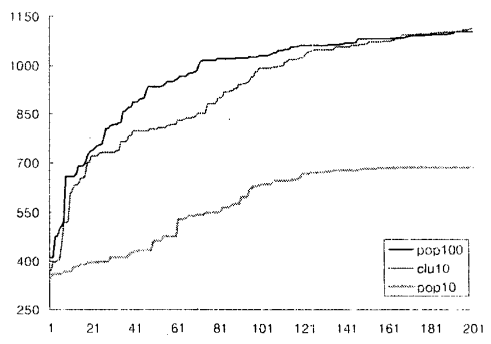

Fig. 11(b) Benchmark result for De Jong function 5

De Jong function 3 is 5 dimensional and De Jong function 5 is 30 dimensional. In Fig. 10(b) we can see several leaps of fitness. The leaps are higher and more frequent in pop100 and clu10 than pop10. Because of the complexity of De Jong function 4, the leap is expressed as a gentle slope in fig. 11(b), but the performance of pop10 is pretty worse than those of pop100 and clu10 as well.

<Function 6> De Jong Function 5

$$f(x)=0.002+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6},$$

$$-65.536\le x_i\le 65.536,$$

$$(a_{ij})=\begin{pmatrix}-32 & -16 & 0 & 16 & 32 & -32 & \cdots & -16 & 0 & 16 & 32\\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32 & 32\end{pmatrix}$$

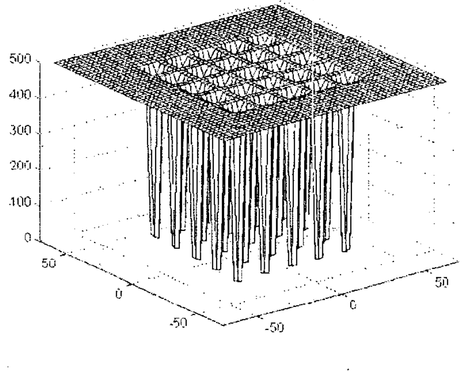

Fig. 12(a) 3D landscape of De Jong function 5

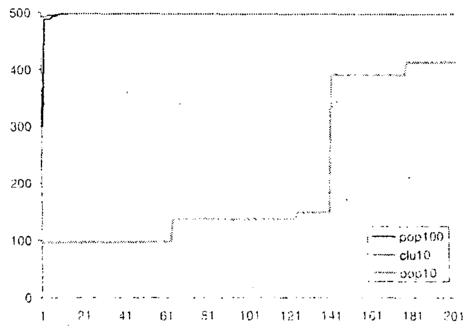

Fig. 12(b) Benchmark result for De Jong function 5

This De Jong function 5 is also called as Shekel's Foxholes function. The landscape has 25 significant holes within a plane, and it is drawn as Fig. 12(a). For this function pop100 and clu10 converges immediately while pop10 shows relatively poor and slow convergence.

<Function 7> Rastrigin Function

$$f(x)=3.0n+\sum_{i=1}^{n}x_i^2-3.0\cos(2\pi x_i)\text{, n=20, } -5.12\le x_i\le 5.12$$

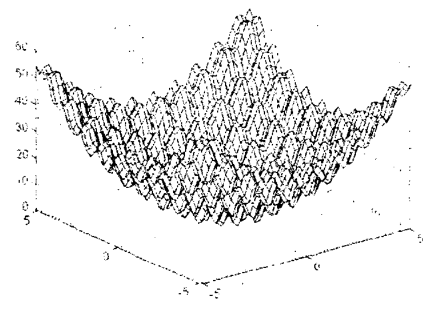

Fig. 13(a) 3D landscape of Rastrigin function

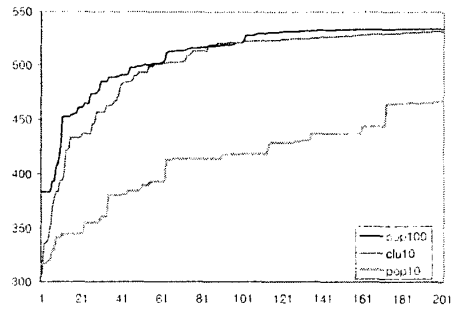

Fig. 13(b) Benchmark result for Rastrigin function

A distinguishing feature of Rastrigin function is that there are so many local solutions and as it gets farther from the global solution, more local solutions are found. Fig. 13(b) shows that pop10 is easily trapped on local optima and hence evolves slower than clu10 and pop100.

<Function 8> Schwefel Function

$$f(x)=418.9829n+\sum_{i=1}^{n}x_i\sin\left(\sqrt{|x_i|}\right),$$

n=10, $-500.0\le x_i\le 500.0$

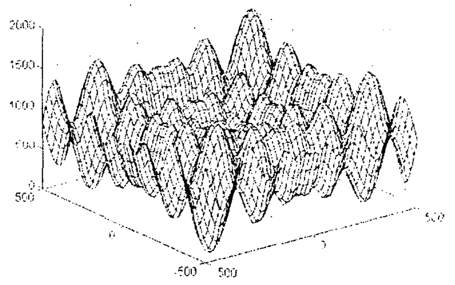

Fig. 14(a) 3D landscape of Schwefel function

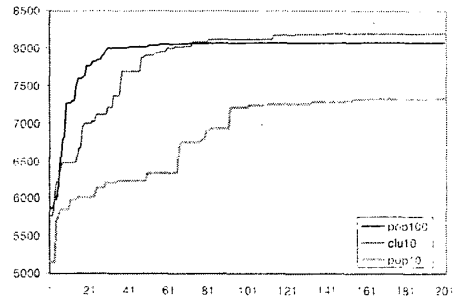

Fig. 14(b) Benchmark result for Schwefel function

The landscape of Schwefel function in Fig. 14(a) shows that there exists the second optimal solution, far from global optimum. This point leads optimization algorithm to be trapped in it. Fig 14(b) shows that pop100 shows the best performance and clu10 evolves a little bit slower than pop100. On the other hand, pop10 shows significantly poor performance.

From the results above we can see that the proposed GA has almost same performance as pop100 though clu10 evaluates only 10 individuals. In every benchmark test clu10 shows rapid convergence as well as pop100. On the contrary pop10, which evaluates 10 times without clustering shows significantly poor performance relative to other algorithms.

Therefore, we can assert that the proposed hybrid GA with clustering has efficiently reduced the evaluation number without any significant loss of the performance.

5 Conclusions

We have proposed an efficient genetic algorithm with less fitness evaluation by clustering. It divides whole population into several clusters, and evaluates one individual for each cluster. The fitness values of other individuals are estimated from the representative fitness values indirectly. This hybrid GA with clustering can efficiently reduces the evaluation number without any loss of the performance. Results from several benchmark show that the algorithm has almost same performance to simple GA that evaluates far more times than the proposed GA.

Such an approach is very useful for problems that require high cost to evaluate individuals. A good example is interactive genetic algorithm application. IGA is one kind of GA, which uses user evaluation as the fitness function. With this 'interaction' IGA can be applied to some special domains such as music or design, different from simple GA. However, human evaluator can become tired easily with repeated interaction, which limits the number of evaluation of IGA application. In this case our proposed GA with clustering can provide the effect of maintaining large population without any additional user evaluation.

There remain several points to improve our research. First, we have to explore the clustering methods and the indirect evaluation methods. They are two important techniques of our algorithm and we have used just one method for each: k-means algorithm for clustering, and Euclidean distance measure for indirect evaluation. Now we are working to substitute them with other clustering algorithms such as SOM, FCM, and isodata algorithm as well as other indirect evaluation methods such as association method, correlation method, and probabilistic method.

After that we hope to apply the algorithm to real IGA problem. But it is not so easy and it requires more research to prove that the proposed hybrid IGA has better performance than original IGA. We are in intensive work for this research.

References

[Anderberg73] Anderberg, M. R., *Cluster Analysis for Applications*, Academic Press, 1973.

[Chamber95] Chamber, L., *Practical Handbook of Genetic Algorithms*, CRC Press, 1995.

[Eberhart96] Eberhart, R., Simpson, P. and Dobbins, R., *Computational Intelligence PC Tools*, Waite Group Press, 1996.

[Fukunaka90] Fukunaka, K., *Introduction to Statistical Pattern Recognition*, Academic Press, 1990.

[Goldberg89] Goldberg, D. E., *Genetic Alrogithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Co. Inc., 1989.

[Gose96] Gose, E., Johnsonbaugh, R. and Jost, S., *Pattern Recognition and Image Analysis*, Prentice Hall PTR, 1996.

[Haritigan75] Haritigan, J. A., *Clustering Algorithms*, John Wiley & Sons, 1975.

[Höppner99] Höppner, F., Klawonn, F., Kruse, R. and Runkler, T., *Fuzzy Cluster Analysis*, John Wiley & Sons, 1999.

[Ingu99] Ingu, T. and Takagi, H., "Accelerating a GA convergence by fitting a single-peak function," *Proc. of IEEE Intl' Fuzzy Systems Conf.*, vol. 3, pp. 1415~1420, 1999.

[Kandel82] Kandel, A., *Fuzzy Techniques in Pattern Recognition*, A Wiley-Interscience Publication, 1982.

[Seront00] Seront, G. and Bersini, H., "A new GA-local search hybrid for continuous optimization based on multi level single linkage clustering," *Proc. of GECCO-2000*, pp. 90~95, 2000.

[Takagi98] Takagi, H., "Interactive evolutionary computation: Cooperation of computational intelligence and human kansei," *Proc. of 5th Int'l Conf. on Soft Computing*, pp. 41~50, 1998.

[Unemi98] Unemi, T., "A design of multi-field user interface for simulated breeding," *Proc. of 3rd AFSS*, pp. 489~494, 1998.

[Xie91] Xie, X. L. and Beni, G., "A validity measure for fuzzy clustering," *IEEE Trans. of Pattern Analysis and Machine Intelligence*, vol. PAMI-13, no. 8, pp. 841-847, 1991.