Domain Decomposition Evolutionary Algorithm for Multi-Modal Function Optimization

Guangming Lin[1], Lishan Kang[2], Yongsheng Liang[1] and Yuping Chen[2]
[1]*Shenzhen Institute of Information Technology, Shenzhen 518029,*
[2]*School of Computer, China University of Geosciences, Wuhan,*
PRC.

1. Introduction

The Simple Genetic Algorithm (SGA) is applied more and more extensively since it was proposed by J. H. Holland [1] in 1970's. SGA is an optimization method based on population by emulating the evolvement disciplinarian of the nature. It has showed the great advantage of quick search for optimal solutions while applied in the optimization of single-modal functions. But as we all know many problems in reality belong to the optimization of multi-modal function, and if SGA is applied to solve this kind of problems, it has the confliction between the search space and convergence speed: the expansion of search space will slow down the convergence speed and the acceleration of convergence speed will reduce the search space, lead to early convergence and as a result stop research at some local optimal solutions.
Evolutionary algorithms have been used regularly to solve multi-modal function optimization problems, due to their population-based approach and their inherent parallelism, *e.g.* a crowding factor model proposed by De Jong[2], a shared-function model proposed by Goldberg and Richardson[3], an artificial immune system method, a split ring parallel evolutionary algorithm, etc., all of which have attempted to maintain the diversity of the population during the process of evolution. In this chapter, we introduce a new 'Domain Decomposition Evolutionary algorithm (called DDEA) which can solve not only simple nonlinear programming problems effectively and efficiently, but can also find the multiple solutions of multi-modal problems in a single run. The DDEA employs dual strategy approach that searches at two levels of detail (namely global then local). In the first (global) step, a Self-adaptive Mutations with Multi-parent Crossover Evolutionary Algorithm (SMMCEA)[4] is employed to perform a global search to divide the (chromosome) population into several subpopulations or niches in subdomains, which is domain decomposition. In the second (local) step, an evolutionary strategy-like algorithm is employed to perform a local search on each isolated niche independently. Then the best solutions of the multi-modal problem are exploited.
The remainder of the chapter is organized as follows. Section 2 introduces a Self-adaptive Mutations with Multi-parent Crossover Evolutionary Algorithm (SMMCEA); Section 3 introduces Domain Decomposition evolutionary algorithm (DDEA); Section 4 presents the successful results of applying DDEA to several challenging numerical multi-modal optimization problems; Section 5 concludes.

Source: Advances in Evolutionary Algorithms, Book edited by: Witold Kosiński, ISBN 978-953-7619-11-4, pp. 468, November 2008, I-Tech Education and Publishing, Vienna, Austria

2. Introduction of SMMCEA

2.1 The Problem to Solve

The general non-linear programming (NLP) problem can be expressed in the following form:

$$\text{Minimize } f(X,Y)$$
$$s.t. \quad h_i(X,Y)=0 \quad i=1,2,\ldots,k_1, \quad g_j(X,Y)\le 0 \quad j=k_1+1, k_1+2,\ldots,k \tag{1}$$
$$X^{lower} \le X \le X^{upper}, \quad Y^{lower} \le Y \le Y^{upper}$$

where $X \in R^p$, $Y \in N^q$, and the objective function $f(X,Y)$, the equality constraints $h_i(X,Y)$ and the inequality constraints $g_j(X,Y)$ are usually nonlinear functions which include both real variable vector X and integer variable vector Y.

Denoting the domain $D = \{(X,Y) \mid X^{lower} \le X \le X^{upper}, \ Y^{lower} \le Y \le Y^{upper}\}$, we introduce the concept of a subspace V of the domain D. m points (X_j,Y_j), $j=1,2,\cdots,m$ in D are used to construct the subspace V, defined as :

$$V = \{(Xv,Yv)\in D \mid (Xv,Yv) = \textstyle\sum_{i=1}^{m} a_i (X_i, Y_i)\}$$

where ai is subject to $\sum_{i=1}^{m} a_i = 1$, $-0.5 \le a_i \le 1.5$.

Because we deal mainly with optimization problems which have real variables and INequality constraints, we assume $k_1 = 0$ and $q = 0$ in the expression (1).

Denoting $w_i(X) = \begin{cases} 0, & g_i(X) \le 0 \\ g_i(X), & \text{otherwise} \end{cases}$ and $W(X) = \sum_{i=1}^{k} W_i(X)$

Then problem (1) can be expressed as follows:

$$\text{Minimize } f(X) \qquad X \in D \tag{2}$$

Subject to

$$W(X)=0 \qquad X \in D$$

We define a Boolean function "*better*" as:

$$better(X_1, X_2) = \begin{cases} W(X_1) < W(X_2) & \text{TRUE} \\ W(X_1) > W(X_2) & \text{FALSE} \\ (W(X_1) = W(X_2)) \wedge (f(X_1) \le f(X_2)) & \text{TRUE} \\ (W(X_1) = W(X_2)) \wedge (f(X_1) > f(X_2)) & \text{FALSE} \end{cases}$$

If *better* (X_1, X_2) is TRUE, this means that the individual X_1 is "better" than the individual X_2.

2.2 Related Work

In 1999, Guo Tao proposed a multi-parent combinatorial search algorithm (GTA) for solving non-linear optimization problems in his PhD thesis [5]. Later it was developed as a kind of subspace stochastic search algorithm [6], that can be described as follows:

```
Guo Tao's Algorithm (GTA)
Begin
    initialize popln $P = \{X_1, X_2, \cdots, X_N\}$;  $X_i \in D$ since (q = 0 implies no integer variables)
        generation count $t := 0$;
        $X_{best} = \arg \underset{1 \le i \le N}{Min} f(X_i)$;
        $X_{worst} = \arg \underset{1 \le i \le N}{Max} f(X_i)$;
    while $abs(f(X_{best}) - f(X_{worst})) > \varepsilon$ do
                select randomly $m$ points $X_1', X_2', \cdots, X_m'$ from $P$ to form the subspace $V$;
                select randomly one point $X'$ from $V$;
                  If better $(X', X_{worst})$ then $X_{worst} := X'$;
                  $t := t + 1$;
                  $X_{best} = \arg \underset{1 \le i \le N}{Min} f(X_i)$;
                  $X_{worst} = \arg \underset{1 \le i \le N}{Max} f(X_i)$
    end do
            output $t, P$;
End
```

where N is the size of population P, $(m-1)$ is the dimension of the subspace V (if the m points (vectors) that construct the subspace V are linearly independent), t is the number of generations, ε is the accuracy of solution. $X_{best} = \arg \underset{1 \le i \le N}{Min} f(X_i)$ means that X_{best} is the variable (individual) in X_i (i=1, 2,$\cdots$, N) that makes the function $f(X)$ have the smallest value. The sub-population in GTA is families which reproduce sexually through the number of m individuals randomly selected from P. The best individual in the sub-population takes part in competition to replace the worst individual in P, therefore the pressure of elimination through selection is minimum. There is no mutation operator, only using multi-parents crossover in GTA.

2.3 A self-adaptive evolutionary algorithm

Since Guo's algorithm deals mainly with continuous NLP problems with Inequality constraints, to make it a truly universal and robust algorithm for solving general NLP problems, we extend Guo's algorithm by adding to it the following improvements:

(1) Guo selected randomly only one candidate solution from the current subspace V. Although he used the concept of a subspace to describe his algorithm, he did not really use a subspace search, but rather a multi-parent crossover. Because he selected randomly only one individual in the subspace, this action would tend to ignore better solutions in the subspace, and hence influence negatively the quality of the result and the efficiency of the search. If however, we select randomly several individuals from the subspace, and substitute the best one for the worst one in the current population, the search should be better. So we replace the instruction line in Guo's algorithm:

"select randomly one point *X′*from *V*; "

with the two instruction lines:

" select randomly *s* points X_1^* , X_2^* , … , X_s^* from *V*;

$$X' = \arg \underset{1 \le i \le s}{Min} f(X_i^*);"$$

(2)The dimension *m* of the subspace in Guo's algorithm is fixed (i.e. *m* parents reproduce). The algorithm always selects a substitute solution in subspaces which have the same dimension, regardless of the characteristics of the solutions in the current population. Thus, when the population is close to the optimal value, the searching range is still large. This would apparently result in unnecessary computation, and affect the efficiency of the search. We can in fact reduce the search range, that is to say, the dimension of the subspaces. We therefore use subspaces with variable dimensions in the new algorithm, by adding the following instruction line to Guo's algorithm:

if $abs\ (f(X_{best}) - f(X_{worst})) \le \eta$ **.and.** $m \ge 3$ **then** $m := m - 1$;

where η depends on the computation accuracy ε, and $\eta > \varepsilon$. For example, if the computation accuracy $\varepsilon = 10^{-14,}$ then we can set $\eta = 10^{-2}$ or $10^{-3.}$

(3) We know in principle that Guo's algorithm can deal with problems containing EQuality constraints. For example, we can use the device of setting two INequality constraints $0 \le h_i(X, Y)$ and $h_i(X, Y) \le 0$ to replace the equality constraint $h_i(X, Y) = 0$, but the experimental results when employing this device are not ideal. However, equality constraints are likely to exist in real-world problems, so we should find methods to deal with them. One such method is to define a new function *W*(*X*, *Y*)

Where $W(X, Y) = \sum_{i=1}^{k} W_i(X, Y)$

$$W_i(X,Y) = \begin{cases} \left|h_i(X,Y)\right|, & i = 1,2,\cdots,k_i \\ \max\{o, g_i(X,Y)\}, & i = k_1+1, k_1+2,\cdots,k. \end{cases}$$

(4) The penalty factor *r* is usually fixed. However, some people use it as a variable, such as Cello[7], who employed a self-adaptive penalty function, but his procedure was rather complex (using two populations). We also make *r* a variable namely *r* = *r* (*t*), where *t* is the iteration count. It can self-adjust according to the reflection information, so we label it a "self-adaptive penalty operator". Since the constraints have been normalized, *r* is relative only to the range of the objective function, which ensures a balance between the errors of the fitness function and the objective function, in order of magnitude.

(5) Guo's algorithm can deal only with continuous optimization problems. It cannot deal directly with integer or mixed integer NLP problems. In our algorithm, when we are confronted with such problems, we need only replace the integer variables derived from the

range of the float of the fitness function with "*integer function*" $int(Y^*)$, where $int(Y^*)$ is defined as the integer part of Y^*. No other changes to the algorithm are needed.

(6) The only genetic operator used in Guo's algorithm was crossover. However, we can add self –adaptive mutations in it, we introduce a better of Gaussian and Cauchy mutation operator into the subspace search. For Gaussian density function f_G with expectation 0; and variance σ^2 is

$$f_G = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{x^2}{2\sigma^2}} \quad , \qquad -\infty < x < +\infty$$

For Cauchy density function f_C with scale parameter $t>0$ is,

$$f_C = \frac{1}{\pi}\,\frac{1}{t^2+x^2} \quad , \qquad -\infty < x < +\infty$$

2.4 A Self-adaptive mutations with multi-parent crossover evolutionary algorithm

Considering the above points, we introduce a new algorithm as follows:

Denoting $Z = (X, Y^*)$, where $Z \in D^*$, and

$D^* = \{(X, Y^*) \mid X^{lower} \le X \le X^{upper},\ Y^{lower} \le Y^* \le Y^u,\ \ X \in R^p,\ Y^* \in R^q\}$, we define integer vector $Y=\text{int}(Y^*)$, where $Y^u = Y^{\text{upper}}+0.999\cdots9I$

Denoting $W(Z) = W(X, \text{int}(Y^*))$,

we define the Boolean function "*better*" as follows:

$$better(Z_1, Z_2) = \begin{cases} W(X_1) < W(X_2) & \text{TRUE} \\ \mathrm{W}(X_1) > \mathrm{W}(X_2) & \text{FALSE} \\ (\mathrm{W}(X_1) = \mathrm{W}(X_2)) \wedge (f(X_1) \le f(X_2)) & \text{TRUE} \\ (\mathrm{W}(X_1) = \mathrm{W}(X_2)) \wedge (f(X_1) > f(X_2)) & \text{FALSE} \end{cases}$$

The general NLP problem (1) can be expressed as follows:

$$\text{Minimize } f(X, \text{int}(Y^*)) \quad \text{in } D^* \quad \text{S.t.} \tag{3}$$

$$\mathrm{W}(Z)=0\,, \qquad Z \in D^*$$

The new algorithm can now be described as follows:

SMMCEA :

```
Begin
    initialize P = {$Z_1, Z_2, \ldots, Z_N$};  $Z_i \in D^*$;
    $t := 0$;
    $Z_{best} = \arg\min_{1 \le i \le N} f(Z_i)$;
```

```
        $Z_{worst} = \arg\max_{1 \le i \le N} f(Z_i)$;
        while not abs ( $F(Z_{best}) - F(Z_{worst})$) ≤ε do
           select randomly $M$ points $Z_1', Z_2', \ldots, Z_M'$ from $P$ to form the subspace $V$;
           select $s$ points randomly $Z_1^*, Z_2^* \ldots Z_s^*$ from $V$;
           for i=1,…s do
              for j=1,…p+q do
              $Z_{Gi}^*(j) := Z_i^*(j) + \sigma_i(j) N_j(0,1)$
              $Z_{Ci}^*(j) := Z_i^*(j) + \sigma_i(j) C_j(1)$
              $\sigma_i(j) := \sigma_i(j)\exp(\tau N(0,1) + \tau' N_j(0,1))$
           endfor
        if better($Z_{Gi}^*, Z_{Ci}^*$) then $Z_i'^* := Z_{Gi}^*$ else $Z_i'^* := Z_{Ci}^*$ ;
        endfor
        $Z' = \arg\min_{1 \le i \le N} f(Z_i)$;
        if better ($Z', Z_{worst}$) then $Z_{worst} := Z'$;
        $t := t + 1$;
        $Z_{best} = \arg\min_{1 \le i \le N} f(Z_i)$;
        $Z_{worst} = \arg\max_{1 \le i \le N} f(Z_i)$;
        if abs ($f(Z_{best}) - f(Z_{worst})$) ≤η .and. $M \ge 3$ then
           $M := M - 1$;
  endwhile
  output $t$, $Z_{best}$, $f(Z_{best})$ ;
end
```

Where Z_{Gi}^* (j), Z_{Ci}^* (j) and σ_i (j) denote the j-th component of the vectors Z_{Gi}^*, Z_{Ci}^* and σ_i, respectively. N(0,1) denotes a normally distributed one-dimensional random number with mean zero and standard deviation one. $N_j(0, 1)$ indicates that the Gaussian random number is generated anew for each value of j. $C_j(1)$ denotes a Cauchy distributed one-dimensional random number with *t=1.*

The factors τ and τ' have commonly set to $\left(\sqrt{2\sqrt{(p+q)}}\right)^{-1}$ and $\left(\sqrt{2(p+q)}\right)^{-1}$.

The new algorithm has the two important features:

1. This algorithm is an ergodicity search. During the random search of the subspace, we employ a "non-convex combination" approach, that is, the coefficients a_i of $Z' = \sum_{i=1}^{m} a_i Z_i'$ are random numbers in the interval [-0.5, 1.5] This ensures a non-zero probability that any point in the solution space is searched. This ergodicity of the algorithm ensures that the optimum is not ignored.

2. The monotonic fitness decrease of the population (when the minimum is required). Each iteration ($t \to t+1$) of the algorithm discards only the individual having the worst fitness in the population. This ensures a monotonically decreasing trend of the values of objective function of the population, which ensures that each individual of the population will reach the optimum.

When we consider the population $P(0)$, $P(1)$, $P(2)$,…, $P(t)$,… as a Markov chain, we can prove the convergence of our new algorithm. See [12].

3. Introduction of DDEA

Experiments indicate that if SMMCEA is directly applied to the optimization of multi-modal function, it is easy to encounter the following two conditions:

1. If keep searching with relatively large population size and crossover size, the individuals of the population will spread around near different modals, but it's difficult for population to get any more improvement and to reach all the modals exactly.
2. If keep searching with relatively small population size and crossover size, the individuals of the population will converge rapidly and reach a few modals, but lose many other modals.

To adopt it to the optimization of multi-modal functions, we combine the above two conditions together and forms two-phase evolutionary algorithm. we divide the optimization procedure into two phases: the first phase is called global optimization, which keeps searching with relatively large population size and crossover size in order to determine the neighborhood of all modals; the second phase is called local optimization, which begins search from each of the neighborhoods which is determined by the global optimization and then keep searching with relatively small subpopulation size and crossover size in order to converge rapidly and reach the modals respectively.

In addition, we introduce the following strategies to make the algorithm suitable to the different tasks of the two phases:

1. During the phase of global optimization, in order to avoid the loss of some obtained modals we introduce the strategy of good individuals isolation: before each evolvement all the individuals in the current population are sorted by their fitness value and then some of the good individuals are limited not to be parents in the next multi-parent crossover.
2. During the phase of local optimization, in order to make all the subpopulations converge to their modals respectively more quickly, we introduce the strategy of best individual exemplar: the best individual of the current population will be compelled to be one of the parents in the next multi-parent crossover.
3. During the phase of local optimization, in order to begin search based on the result of the global optimization and to keep the search around the neighborhood of all the modals, to each modal we will construct a local feasible area η, which is to be modified during the evolvement.

The detailed procedures of the optimization DDEA are as the following:

Phase 1: Global optimization (using SMMCEA)

```
Randomly initialize population $P(0)= \{P_1,P_2,\ldots, P_{N1}\}$,Evaluate $P(0)$,$t_1=0$
while $t_1< MAXT_1$ do
  randomly select $m_1$ parents from $P(t_1)$ with the strategy of good individuals isolation
  produce a child by multi-parent crossover and self-adaptive Gaussian and Cauchy
  mutation
    if the child is better than the worst individual of $P(t_1)$ then
          replace the worst individual of $P(t_1)$ with the child
          end if
          $t_1= t_1+1$
end while
```

Phase 2: Local optimization

```
     for $k= 1$ to $N_1$ do
       initialize local feasible area $\eta$, which is the rectangle area around $P_k$ with the
       radium r
       Randomly initialize subpopulation SUBP(0) within the area of $\eta$
                 $SUBP(0)=\{ SUBP_1, SUBP_2,\ldots, SUBP_{N2} \}$
       $t_2=0$
       while ($t_2< MAXT_2$ and individuals of $SUBP(t_2)$ are different )do
            randomly select $m_2$ parents from $SUBP(t_2)$ with the strategy of the best
         individual exemplar
                produce a child by multi-parent crossover
                if the child $\in \eta$ and it is better than the worst individual of $SUBP(t_2)$
                then replace the worst individual of $SUBP(t_2)$ with the child
                end if
                     evaluate the best individual of $SUBP(t_2)$, which is named as
                     $SUBP_{best}$
                modify local feasible area $\eta$, make it as the rectangle area around
                $SUBP_{best}$
         with the radium r
                     $t_2= t_2+1$
    end while
    output the best individual of $SUBP(t_2)$
end for
```

The new algorithm employs a zoomed (global to local) dual strategy (two steps) approach. The first (global) step employs a global search, *i.e.* it divides the (chromosome) population into L $(L \le k)$ niches, each of which includes at least one of the k optimal solutions (if the objective function is continuous in D^*). This step uses a SMMCEA [4]. If the number of parents M in the multi-parent recombination operator is large enough, for example, $M \ge 8$,

then after sufficient large generations the population is decomposed into subpopulations (each of which approaches to an optimal solution), else it will converge to only one solution [11].
The second (local) step employs an evolution strategy [13] to search for the local optima in the chosen *L* subspaces determined by the subpopulations. Since the *L* optimal solutions are located in separate subspaces, the local strategy consists of two sub-steps:
a). Rank the individuals of the population obtained from the first (global) step according to their fitness values. Then choose the best *L* individuals from the population, ensuring that they are not close to each other like hedgehogs.
b). Generate *L* subspaces with the chosen individual at the center of each. Search these niches locally until each subspace converges to an optimal solution. If one does not know how many optimal solutions a given problem has, one can predict the number *k*, for example, by using the number of individuals whose fitness values are larger than the average fitness value.
The algorithm has different limiting behaviors for different problems, namely:
a). When the problem has only $k = 1$ solution, *i.e.* the only globally optimal solution. Following the nature of population descent, all of the individuals will descend together to the bottom of the valley.
b). When the problem has $k > 1$ solutions, *i.e.* if $k \le N$, where *N* is the size of the population, *k* solutions may be generated in the population. The algorithm will then find multi-solutions in a single run.

4. Numerical experiments and analysis

Example 1 Humpback function (the function has six local optimal solutions, two of which are global optimal solutions)

$$\min f(x_1, x_2) = (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$$

where $x_1 \in [-3,3], x_2 \in [-2,2]$

Example 2 Typical function with many global optimal solutions(the function has increasing number of global optimal solutions while *j* is increased)

$$\min f(x_1, x_2) = 3 - (\sin(jx_1))^2 - (\sin(jx_2))^2,$$

where $x_1, x_2 \in [0,6], j = 1,2,\cdots$

Example 3 Absolute value function(the function has a plenty of local optimal solutions, 16 of which are global optimal solutions)

$$\min f(x_1, x_2) = \prod_{i=1}^{4} | x_1 - 3i | + \prod_{j=1}^{4} | x_2 - 4j |,$$

where $x_1 \in [0,13], x_2 \in [0,17]$

Example 4 N-dimension Shubert function[8](when n=2, the function has 720 local optimal solutions, 18 of which are global optimal solutions)

$$\min f(x_1, x_2, \cdots, x_n) = \prod_{i=1}^{n} \sum_{j=1}^{5} j \cos((j+1)x_i + j)$$

where $x_i \in [-10, 10], i = 1, 2, \cdots, n$

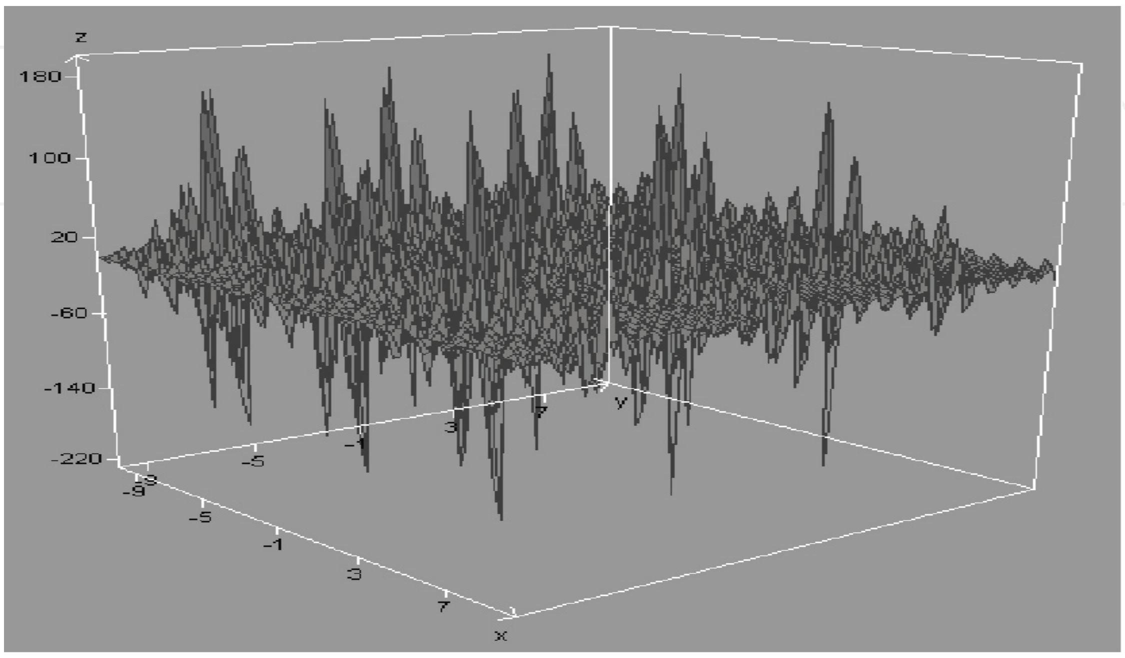


Fig. 1. Shubert function

All the examples mentioned above are representatives of different kinds of functions. ***Example 1***, ***Example 2*** and ***Example 4*** are cited from [9]. ***Example 1*** is the representative of glossy function with only a few modals, ***Example 2*** is the representative of glossy function with many modals, ***Example 3*** is the representative of non-glossy function, and example 4 is the representative of high-dimension function. Generally we can get satisfying optimal solutions when we set the parameters according to the following principle:
The phase of global optimization: $N_1 \approx 10$*the number of actual optimal solutions

$$2000 < MAXT_1 < 100*N_1$$

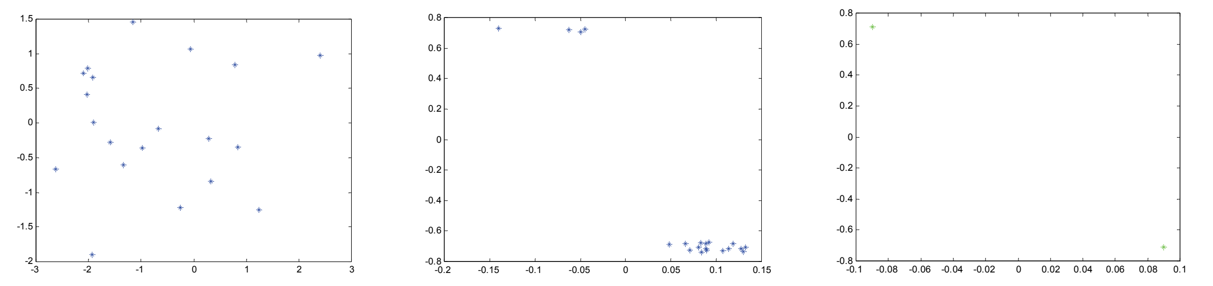
$$6 \le m_1 \le 10$$

The phase of local optimization: $10 < N_2 < 20, r = 2.0$

$$2000 < MAXT_2 < 5000$$

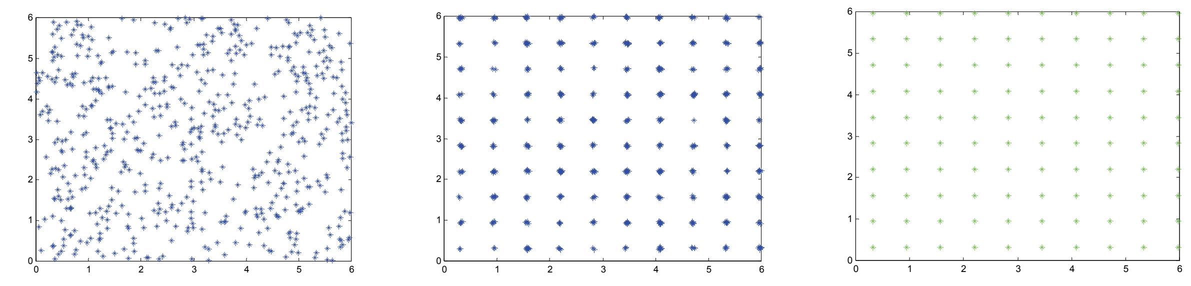
$$3 \le m_2 \le 5$$

The following figures show population distribution in different phases for each example, which indicate the optimization procedures of different examples. Each figure has three parts: (a) is the distribution of population after randomly initialization; (b) is the distribution of population after global optimization; (c) is the distribution of the found modals after local optimization. The horizontal coordinate is the value of x_1 and the vertical coordinate is the value of x_2.



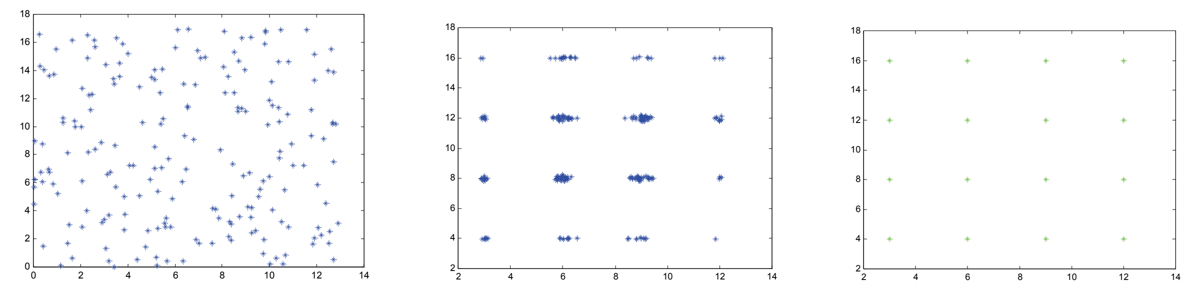
(a) after randomly initialization (b) after global optimization (c) after local optimization

Fig. 2. Population distribution for example 1



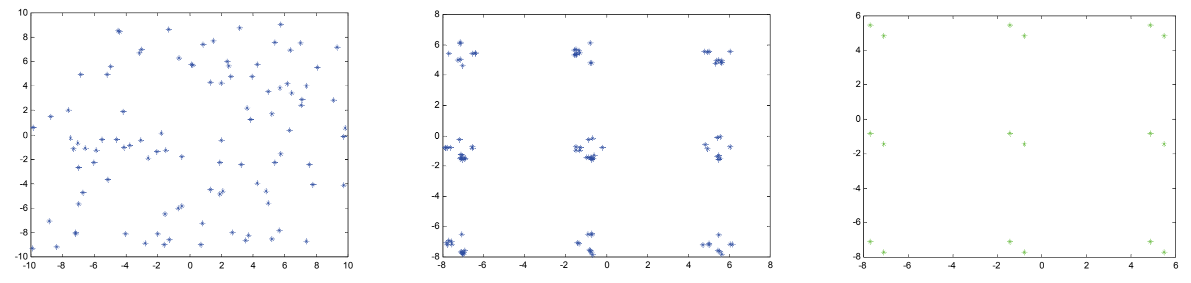
(a) after randomly initialization (b) after global optimization (c) after local optimization

Fig. 3. Population distribution for example 2 when j=5



(a) after randomly initialization (b) after global optimization (c) after local optimization

Fig. 4. Population distribution for example 3



(a) after randomly initialization (b) after global optimization (c) after local optimization

Fig. 5. Population distribution for example 4 when n=2

Additionally, the following tables list parameters and results for different experiments:

Example No.	Parameters		results		
	N_1	$MAXT_1$	Actual modals	Found modals	fitness of

					all modals
Example 1	20	2000	2[9]	2	-1.031628
Example 2（j=5）	600	60000	100[9]	100	1.000000
Example 3	200	20000	16	16	0.000000
Example 4（n=2）	100	10000	18[9]	18	-186.730909

Table 1. Experiment parameters and results for each example (other parameters are: m_1=7,m_2=5,N_2=10,$MAXT_2$=2000)

The value of j	2	3	4	5	6	7	8	9	10
Actual modals	16	36	64	100	121	169	225	289	361
Found modals	16	36	64	100	121	169	225	289	361

Table 2. Experiment results for example 2 with different value of j (The fitness of all modals is 1.000000)

Example No.	Parameters		Results	
	N_1	$MAXT_1$	Found modals	fitness of all modals
Example 4（n=3）	800	500000~1000000	81	-2709.09350

Table 3. Parameters and experiment results for example 4 when n=3 (other parameters are: m_1=7,m_2=5,N_2=10,$MAXT_2$=10000).

From population distribution of the optimization procedures showed in Fig2, Fig3, Fig4 and Fig5, as well as the experiment results showed in Tables 1 and Table 2, we can see that DDEA is very efficient for the optimization of low- dimension multi-modal function, usually we can reach all the modals exactly. But Table 3 indicates that when the dimension of the function is increased to higher than two, the efficiency is decreased because of the search space is expanded sharply.

5. Conclusion

We here proposed some self-adaptive methods to choose the results of Gaussian and Cauchy mutation, and the dimension of subspace. We used the better of Gaussian and Cauchy mutation to do local search in subspace, and used multi-parents crossover to exchange their information to do global search, and used the worst individual eliminated selection strategy to keep population more diversity.

Judging by the results obtained from the above numerical experiments, we conclude that our new algorithm is both universal and robust. It can be used to solve function optimization problems with complex constraints, such as NLP problems with inequality and (or) equality constraints, or without constraints. It can solve 0-1 NLP problems, integer NLP problems and mixed integer NLP problems. When confronted with different types of problems, we don’t need to change our algorithm. All that is needed is to input the fitness function, the constraint expressions, and the upper and lower limits of the variables of the problem. Our algorithm usually finds the global optimal value.

In the paper we analyze the character of the multi-parent genetic algorithm, when applied to solve the optimization of multi-modal function, MPGA works in different forms during different phases and then forms two-phase genetic algorithm. The experiments indicate that DDEA is effective to solve the optimization of multi-modal function whose dimension is no higher than two, but to high-dimension function, the efficiency is not eminent and it needs to be improved much more.

6. Acknowledgements

This work was supported by the National Natural Science Foundation of China (No.60772163) and the Natural Science Foundation of Hubei Province (No. 2005ABA234). Thanks especially give to the anonymous reviewers for their valuable comments.

7. References

Holland J H. Adaptation in Natural and Artificial System. *Ann Arbor: The University of Michigan Press*, 1975.

De Jong, K. A. An analysis of the behavior of a class of genetic adaptive systems. *Ph.D. Thesis*, Ann Arbor, MI:University of Michigan. Dissertation Abstracts International, Vol. 36(10), 5410B (University Microfilms No.76-9381).

Goldberg, D. E. and Richardson, J. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*. 41–49.

Guangming Lin, Lishan Kang, Yuping Chen, Bob McKay and Ruhul Sarker, *A self-adaptive Mutations with Multi-paretn Crossover Evolutionary Algorithm for Solving Function Optimization Parbolems Lecture Notes in Computer Science* 4683, pp.157-168, Springer-Verlag Berlin Heidelberg.

Tao Guo, Evolutionary Computation and Optimization. PhD thesis, Wuhan University, Wuhan,1999.

Tao Guo, Kang Lishan. A New Evolutionary Algorithm for Function Optimization. *Wuhan University Journal of Natural Sciences*, 1999, 4(4): 404-419.

Carlos A. Coello: Self-adaptive penalties for GA-based optimization, in Proceedings of the Congress on Evolutionary Computation, Washington, D.C USA, IEEE Press, 1999,537~580

Toyoo Fukuda, Kazuyuki Mori and Makoto Tsukiyama. (1999). Parallel search for multi-modal function optimization with diversity and learning of immune algorithm. In: D. Dasgupta (Ed.) *Artificial Immune Systems and Their Applications*. Springer-Verlag, Berlin, Heidelberg, 210–220.

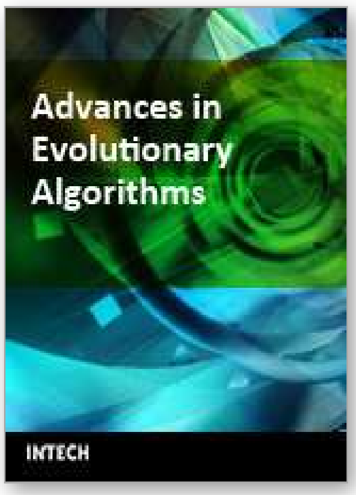
Zhai Hai-feng. Zhao Ming-wang. A Cell Excluse Genetic Algorithm for Finding All Globally Optimal Solutions of Multimodal Function. *Control and Decision*. 1998,13(2):131-135.

Yan Li, Lishan Kang, Hugo de Garis, Zhuo Kang and Pu Liu. (2002).A robust algorithm for solving the nonlinear programming problems. *International Journal of Computer Mathematics*, 79(5), 523–536.

Lishan Kang,Yan Li, Zhuo Kang, Pu Liu andYuping Chen. (2000).Asynchronous parallel evolutionary algorithm for function optimization. 16th world computer congress. *Proceedings of Conference on Software: Theory and Practice*. Electronic Technology Press, Aug., Beijing, 737–742.

Jun He, Lishan Kang. On the convergence rates of genetic algorithms. Theoretical Computer Science, 229 (1999) 23~29

Yan Liexiang and Ma Dexian. (1999). The sequence competition algorithm for global optimization of continuous variables function. *Journal of Hubei Technology College*, 14(1–2).



Advances in Evolutionary Algorithms
Edited by Xiong Zhihui

ISBN 978-953-7619-11-4
Hard cover, 284 pages
Publisher InTech
Published online 01, November, 2008
Published in print edition November, 2008

With the recent trends towards massive data sets and significant computational power, combined with evolutionary algorithmic advances evolutionary computation is becoming much more relevant to practice. Aim of the book is to present recent improvements, innovative ideas and concepts in a part of a huge EA field.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Guangming Lin, Lishan Kang, Yongsheng Liang and Yuping Chen (2008). Domain Decomposition Evolutionary Algorithm for Multi-Modal Function Optimization, Advances in Evolutionary Algorithms, Xiong Zhihui (Ed.), ISBN: 978-953-7619-11-4, InTech, Available from:
http://www.intechopen.com/books/advances_in_evolutionary_algorithms/domain_decomposition_evolutionary_algorithm_for_multi-modal_function_optimization