The Eighth International Symposium on Operations Research and Its Applications (ISORA'09)
Zhangjiajie, China, September 20–22, 2009

An Efficient Asynchronous Parallel Evolutionary Algorithm Based on Message Passing Model for Solving Complex Nonlinear Constrained Optimization

Hao Wu[1] Chunlin Xu[2] Xiufen Zou[2,*]

[1]School of Information Engineering, East China Jiaotong University, Nanchang , 330013, China
[2]School of Mathematics and Statistics, Wuhan University, Wuhan, 430072, China

Abstract This study presents an asynchronous parallel evolutionary algorithm based on message passing model (MAPEA) for solving complex function optimization problems with constraints. The MAPEA combines a local search into the global search. The local search is based on Tabu search, and the radius of neighborhood is self-adaptive. The MAPEA is implemented in Parallel Virtual Machine (PVM) programming environment and used to solve two widely applied complex optimization problems. The speedup and parallel efficiency of MAPEA are analyzes and comparisons with other published results are made. Numerical experiments show that MAPEA exhibits good performance and can handle complex constrained optimization problems.

Keywords Evolutionary algorithm; Asynchronous Parallel; Tabu search; function optimization.

1 Introduction

Real-world optimization problems are often complex and difficult to solve. For example, The "BUMP" function, which developed by Keane in engineering design[1], has been considered as a standard benchmark for nonlinear constrained optimization, because it is highly multi-modal and its optimum is located at the nonlinear constrained boundary and its true maximum is unknown. Various hybridized genetic algorithms and parallel evolutionary algorithms are proposed to solve this problem and some good results are obtained [2-6]. However, these results can be further improved. In this paper, we suggest an asynchronous parallel evolutionary algorithm based on message passing model (MAPEA) for solving complex constrained optimization problems and present detailed comparison for different situations.

2 An Asynchronous Parallel Evolutionary Algorithm Based on Message Passing Model

* Corresponding author: xfzou@whu.edu.cn

In this paper, an Asynchronous Parallel Evolutionary Algorithm Based on Message Passing Model (MAPEA) is presented. Local search based on Tabu search is employed in the algorithm. The details of MAPEA is given as follows:

Algorithm MAPEA:
t=0;
Initialize the population $P(t) = \{P_1, P_2, \dots, P_N\}$.
Evaluate P(t);
While stop_criterion not satisfied do
  If ($t \bmod T_1 = 0$) and (any message received) then
      X_{son} = Receive () (Operator 1)
  Else then
      X_{son} = Multi_crossover () (Operator 2)
  If (Xson is better than X_{worst})
      Then $X_{worst} = X_{son}$
  If (X_{best} does not change after T_2 cycles) then
      X_{best} = Tabu_Search (X_{best}) (Operator 3)
  If ($t \bmod T_3 = 0$) then
      Broadcast X_{best} to other processes (Operator 4)
  t=t+1
end while
end

The flowchart of MAPEA is given in Fig 1.

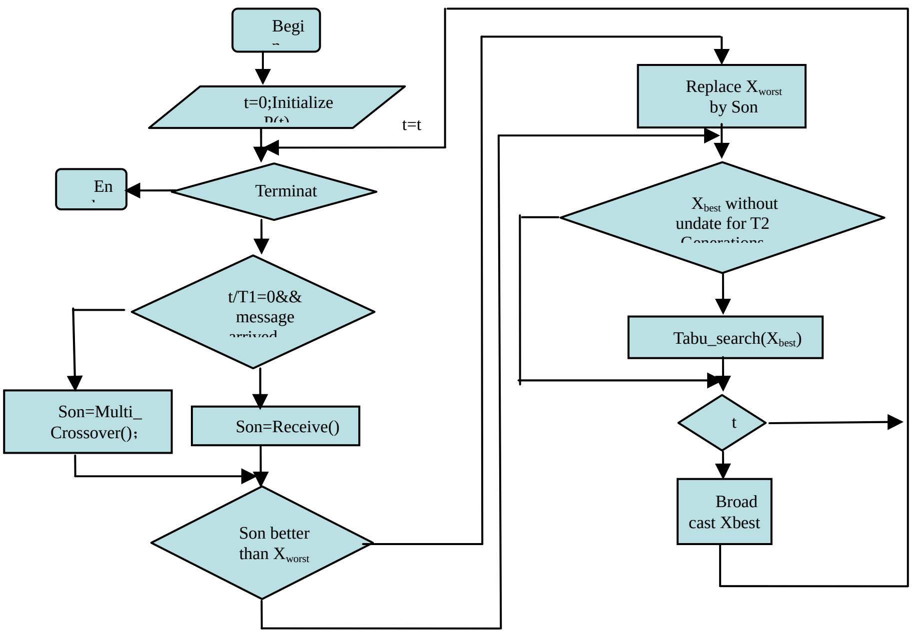

Figure 1: The flowchart of MAPEA

In our program, the stopping criterions are when the number of evolution is greater than the given maximum Generation_max, or the difference of the best and worst fitness is small enough.

(1) Operator 1 and Operator 4

MAPEA uses PVM library functions to handle message passing. Operator 1 and Operator 4 can be performed by using PVM library functions pvm_probe(), pvm_recv(), pvm_upkdouble(), pvm_upkint(), and pvm_initsend(), pvm_pkdouble(), pvm_pkint(), pvm_mcast(), pvm_send() respectively.

(2) Operator 2: Multi_crossover () [2]

Let Son be the convex combination of L parents. $Son = \sum_{i=1}^{L} a_i X_i^p$, where: $\sum_{i=1}^{L} a_i = 1$ ， $-0.5 \le a_i \le 1.5$ ， $i = 1, 2, \cdots L$

(3) Operator 3: Tabu_search() [7,8]

The algorithm is described in the following.

Begin
t:=0；
Give a current solution X^* and let $X_{best} = X^*$；
While（stop_criterion not satisfied)
Generate N_candi candidates $\{X_1, X_2, \cdots, X_{N\_candi}\}$ in neighborhood of X^*;
Evaluate candidates, and rank them from better to worse: $\{X'_1, X'_2, \cdots, X'_{N\_candi}\}$；
If(X'_1 better than X_{best})
$X_{best} = X^* = X'_1$， Add X'_1 to the Tabu List；
Else For(i＝2 to N_candi)
If（ X'_i is not in the Tabu List）
$X^* = X'_i$, Add X'_i to the Tabu List；break;
End while;
End;

While: N_candi candidates are generated in neighborhood of X^* with radius *r*. *r* is set to be self-adapted in this paper. First, let $r = R$, while R is a small positive number. Then,

$$r = \begin{cases} r(1+\alpha) & \text{X}_{\text{best}} \text{ didn't update in last cycle} \\ r/(1+\alpha) & \text{X}_{\text{best}} \text{ update in last cycle} \\ R & \text{X}_{\text{best}} \text{ haven't update for many cycles} \end{cases}$$

Add X'_1 to the Tabu List means, X'_1 cannot be searched in the following List_Length cycles，List_Length is called Tabu List length or the max Tabu Cycles. The Tabu times of any solution in Tabu List is reduced by 1 every cycle. The method to judge whether a solution X is in the Tabu List is to check whether the distance between X and all list elements is smaller than a given positive number Δ. The stopping criterion is when the number of cycles is greater than the given maximum Gen_Tabu. N_candi , r, Gen_Tabu, List_Length and Δ are important parameters in Tabu search.

3 Numerical Experiments and Results

3.1 Two test problems

(1) Bump function[1]

$$Max\ F1(\vec{x}) \equiv \frac{\left|\sum_{i=1}^{n}\cos^4(x_i) - 2\prod_{i=1}^{n}\cos^2(x_i)\right|}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \tag{1}$$

subject to: $g_1(x) = \prod_{i=1}^{n} x_i \geq 0.75, g_2(x) = \sum_{i=1}^{n} x_i < 7.5n$, where:

$$0 \leq x_i \leq 10\ (1 \leq i \leq n)$$

$$(2) Min\ F2(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3-10)^2 + 4(x_4-5)^2 + (x_5-3)^2$$
$$+2(x_6-1)^2 + 5x_7^2 + 7(x_8-11)^2 + 2(x_9-10)^2 + (x_{10}-7)^2 + 45 \quad [9]$$

subject to:

$$105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geq 0$$
$$-3(x_1-2)^2 - 4(x_2-3)^2 - 2x_3^2 + 7x_4 + 120 \geq 0$$
$$-10x_1 + 8x_2 + 17x_7 - 2x_8 \geq 0$$
$$-x_1^2 - 2(x_2-2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \geq 0$$
$$8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0$$
$$-5x_1^2 - 8x_2 - (x_3-6)^2 + 2x_4 + 40 \geq 0$$
$$3x_1 - 6x_2 - 12(x_9-8)^2 + 7x_{10} \geq 0$$
$$-0.5(x_1-8)^2 - 2(x_2-4) - 3x_5^2 + x_6 + 30 \geq 0$$

where: $-10.0 \leq x_i \leq 10.0$, $\left(i = 1, 2, \cdots 10\right)$.

The Handle of constraints uses the Better function [10]. The algorithm is performed on a simulated parallel environment consisting of two PCs with CPU Intel Pentium Dual 3.4GB connected by a 10Mbps Ethernet, and is implemented by PVM

3.4.3.

3.2 Comparison of results under different parameters

Fig.2 and Fig.3 show the evolution of best fitness with the number of generations for functions F1 (n=100) and F2 respectively solving by single process MAPEA with/without Local search. The Tabu search parameters N_candi, r, Gen_Tabu, List_Length, Δ are set to be 20, 0.01,100, 5, $r/20$ respectively. The dashed lines in these two figs clearly show the good performance of MAPEA with Local search.

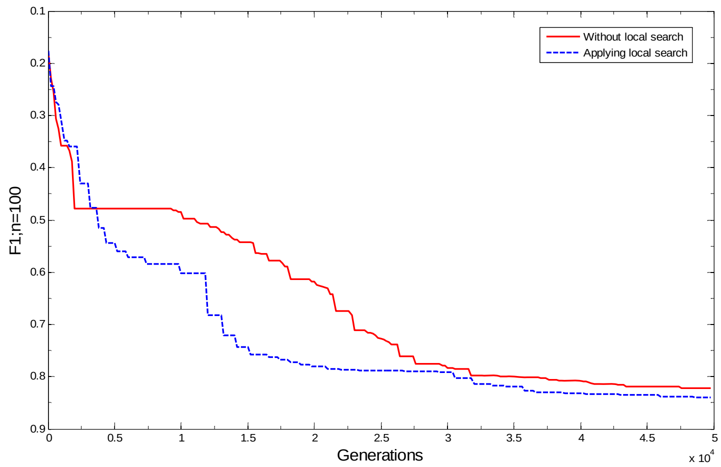

Figure 2: The evolution of the best fitness with the number of generations for test function F1(n=100) with/without Local search

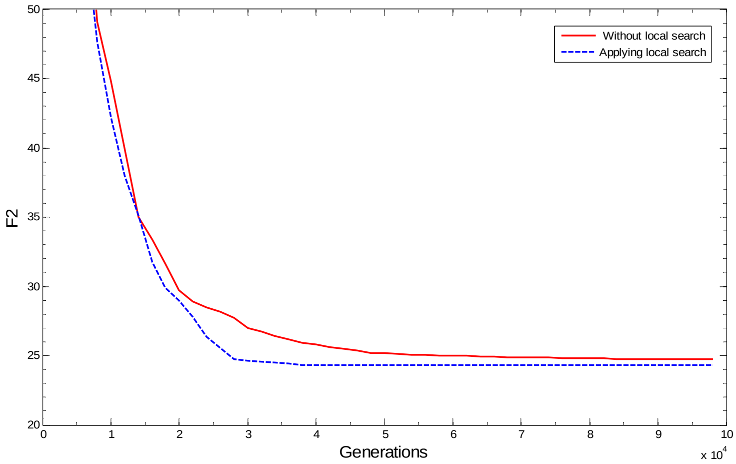

Figure 3: The evolution of the best fitness with the number of generations for test function F2 with/without Local search

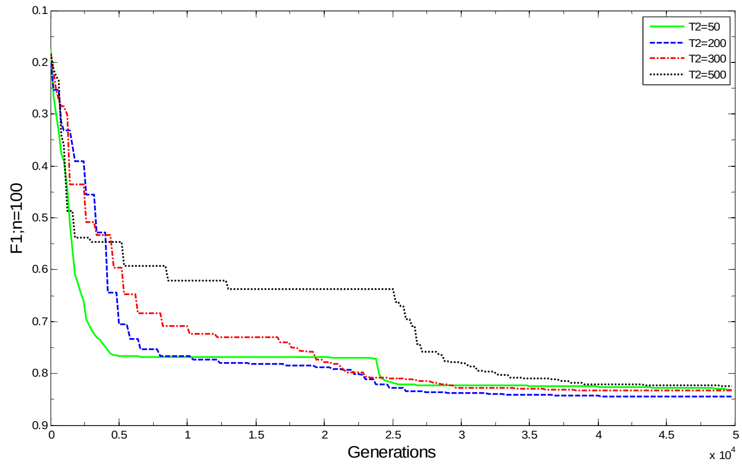

Figure 4: The evolution of the best fitness with the number of generations for test function F1(n=100) with different T2 values

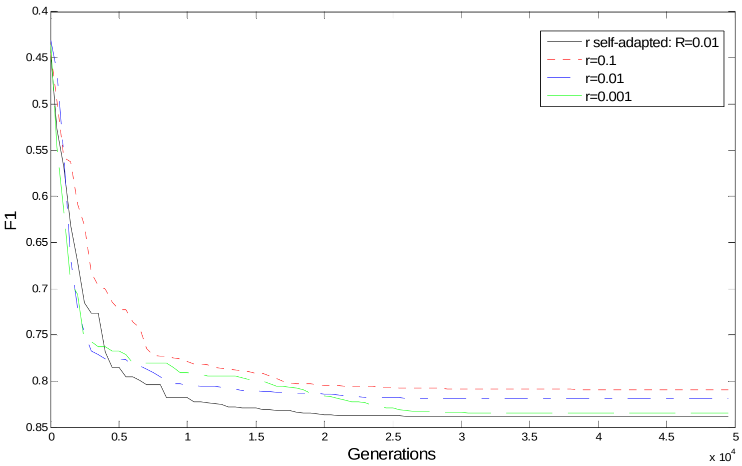

Figure 5: Comparison between self-adapted r and fixed r (F1;n=100)

Threshold value T_2 control the using frequency of Local search. Fig.4 gives the curve of F1(n=100) solving by single process MAPEA under different T_2 values. Tabu search parameters are the same with Fig.2 and Fig.3. The green solid line shows that small T_2 values, ie. MAPEA with High Local search using frequency can fast convergence at early stage but it failed to find a good global optimization at last. The black point line shows a large T_2 values, ie. MAPEA with low using frequency of Local search is not good in both convergence speed and the global optimization ability, while a middle T_2 values can get the best performance.

To test the effect of self-adapted *r*, a comparison between fixed *r* and self-adapted *r* has been given in Fig.5 and Fig.6. In Fig. 5, we set three values: 0.001，0.01，0.1 for fixed *r*, and set R=0.01 in self-adapted *r*. Other parameters such as N_candi , Gen_Tabu, List_Length, Δ are set to be 20, 100, 5, *r*/20 respectively. In Fig. 6, we set three values: 0.0001，0.001，0.01 for fixed *r*, and R=0.01 for self-adapted *r*. N_candi , Gen_Tabu, List_Length, Δ are set to be 20, 200, 5, *r*/20 respectively.

From Fig.5 and Fig.6, we can see that MAPEA shows different features with different *r* values, while it is not so easy to find a good value *r*. Compared to fixed *r*, self-adapted *r* can stably convergent to better solutions.

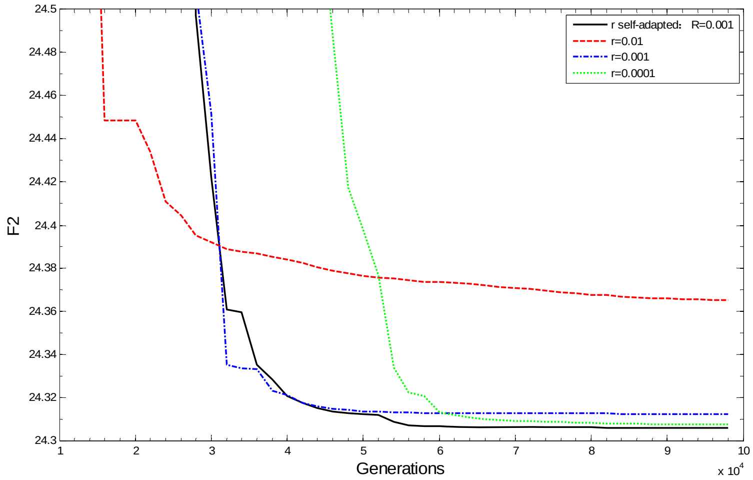

Figure 6: Comparison between self-adapted r and fixed r (F2)

Table 1: Parallel's improving on solution quality

Dimension	one process	two processes	four processes
10	0.74731036152612107	0.74731036152612107	0.74731036152612107
20	0.80361910412558879	0.80361910412558879	0.80361910412558879
50	0.83526220657660477	0.83526222888379931	0.83526222917634629
100	0.84259323041536904	0.84259323060953262	0.84259773783740766
200	0.83252323699857000	0.84465905042905798	0.84692428112721241
300	0.84267960740810217	0.84712574175509692	0.84996186695570108

Table 2: Results of single process

Time(s)	No. of evaluations
7.2215	119473.7

3.3 Analysis of Parallel Efficiency

This section will discuss the improvement of time and solution quality when the apgorithm is implemented by parallel.

3.3.1 Improvement on the quality of optimal solutions

Comparisons in solution quality with single process and multi processes APEA are presented in this Section. Table 1 shows the increasing of process's number can make MAPEA convergent to better solutions, this is quite obviously when the dimension of decision variable increases.

3.3.2 Improvement on running time

The stop_criterion in this section is when a certain solution is obtained. The test problem in this section is Bump function F1(n=100). Table 2 gives the running time and No. of evaluations of single process MAPEA. Table 3 and Table 4 give the running time and No. of evaluations under different T_1 and T_3 values of two and four processes MAPEA respectively. All the data in Table 2-4 is the average value of ten runs.

Table 3: Results of two processes

Case	T_1	T_3	Time(s)	No. of evaluations	Speedup	Efficiency
1	2	200	79.5465	65181.5	0.090783	0.045392
2	10	200	13.62075	72466	0.530184	0.265092
3	50	200	4.144	72282	1.74264	0.87132
4	100	200	4.37825	83531.5	1.649403	0.824702
5	200	200	4.394	101458.5	1.643491	0.821746
6	25	50	5.98425	37319	1.206751	0.603376
7	10	50	13.9095	66026	0.519178	0.259589
8	200	400	4.0895	105344	1.765864	0.882932
9	100	400	3.83925	82602.5	1.880966	0.940483

Notes: Threshold value T_1 control the accept frequency. The accept frequency is high while T_1 is small. Threshold value T_3 control the accept frequency. The accept frequency is high while T_3 is small.

Table 4: Results of four processes

Case	T_1	T_3	Time(s)	No. of evaluations	Speedup	Efficiency
1	2	200	36.404	102681	0.198371	0.049593
2	10	200	11.396	117848.5	0.633687	0.158422
3	50	200	3.148	153955.5	2.293996	0.573499
4	100	200	1.593125	66858	4.532915	1.133229
5	200	200	3.589375	184201	2.01191	0.502978
6	25	50	3.396	157618	2.126472	0.531618
7	10	50	5.270875	68958	1.370076	0.342519
8	200	400	2.53275	155321	2.851249	0.712812
9	100	400	3.67725	219246.5	1.963832	0.490958

Notes: threshold value T1 control the acceptance frequency. The accept frequency is high while T1 is small.; threshold value T3 control the accept frequency. The accept frequency is high while T3 is small.

It is obvious that communication frequency should not be too high. The increasing of communication frequency is helpful to the exchange of information among processes, it makes the algorithm convergent in less iterations, and the number of evaluations is decreased correspondingly. However the running time do not be reduced (Case 1,2,6,7 in Table 3 and 4). This is because the time consumption of the algorithm on information exchange is greatly increased as the increase of communication frequency.

Table 3 and Table 4 indicate that the algorithm appears excellent when T3 is 2-4 times larger than T1, the algorithm gets good speedup and high efficiency at this situation. This is because the experimental environment of this paper is two PCs and each process can receive messages from other processes. So, it will be appropriate that the acceptance frequency is a little higher then sending frequency.

It is exciting that MAPEA reaches super-linear speedup in case 5 at Table 4. It shows that our MAPEA exhibits good performance.

3.3.3 Comparison of solutions with other published literature

Table 5: Comparison of MAPEA and other published algorithms for Bump function (F1)

Dimension	$F1(x^*)$ in Ref[2,3]	$F1(x^*)$ get by MAPEA
10	0.74731036152611	0.74731036152612107
20	0.803619104125588124	0.80361910412558879
50	0.8352620128794	0.83526222917634629
100	0.8448539	0.84259773783740766
200	0.8468442	0.84692428112721241
300	0.8486441	0.84996186695570108

Table 5 gives the comparison of optimal solutions obtained by MAPEA with some published algorithms. It is shown in Table 5 that MAPEA obtains better results except in the case of dimension n=100. Because the algorithm in Ref[3] was implemented on a MPP supercomputer YH-4 and MAPEA on 4 processes, MAPEA has great potential ability to solve complex problems.

When dimensions of decision variables for Bump function F1 are n=50, the obtained optimal solutions are given as follows.

n=50, the optimal solution F1(x*)=0.835262229176346290, where g_1(x)=0.75000000960679236, g_2(x)=78.000534744033104. x* = {6.28374894436865810, 3.16996935340582200, 3.15600772401839880, 3.14240685076789640, 3.12846986723756080, 3.11544735250180600, 3.10181780654239250, 3.08863941435730860, 3.07506648365465330, 3.06175355081532350, 3.04848191683316430, 3.03514686349909100, 3.02162665311784910, 3.00801582807512080, 2.99428134092484300, 2.98094508654189030, 2.96649155065091290, 2.95230776862908590, 2.93794420913259510, 2.92339726443855690, 0.48807066399310273, 0.48595033537392412, 0.48390650166368443, 0.48106670850382371, 0.47966168737935011, 0.47719502002351144, 0.47544818653611809, 0.47346981705730362, 0.47106116281709581, 0.46991189777182168, 0.46686598509958560, 0.46561190892610549, 0.46382014740226279, 0.46168661869414396, 0.45964115919871351, 0.45833006108799790, 0.45709276814929234, 0.45467123449175850, 0.45317800125395846, 0.45111762479357204, 0.44974885361226380, 0.44886340877218944, 0.44667530065435884, 0.44507122170000923, 0.44367479889036610, 0.44230701232669190, 0.44065799208960121, 0.43958801925021213, 0.43783243382101700, 0.43639238318636375}.

When compared with the optimal solution $F2(x^*) = 24.3062090683032$ in Ref[3,10], MAPEA obtains $F2(x^*)=24.306209068179751$, where $x^* =$ {2.17199637169672320, 2.36368297256372670, 8.77392573819491870, 5.09598448742434410, 0.99065476663312746, 1.43057398334893660, 1.32164420904017520, 9.82872580861278070, 8.28009166382506830, 8.37592664533950430}

4 Discussion and Conclusions

This paper presents an asynchronous parallel evolutionary algorithm based on message passing model (MAPEA) for solving complex function optimization problems with constraints and uses two benchmarks to test performance of MAPEA. Because the true maximum of Bump function for different dimensions are unknown, we give the values of objective function, constraints and decision variables are given within 50 dimensions of decision variables by using MAPEA. These works can provide the useful information for further research.

References

[1] A. J. Keane, Experiences with optimizers in structural design, in Proceedings of the Conference on Adaptive Computing in Engineering Design and Control 94, ed. I. C. Parmee, Plymouth(1994),14-27

[2] T. Guo and L. S. Kang, A New Evolutionary Algorithm for Function Optimization [J]. Wuhan University Journal of Natural Sciences, 4(4). (1999), 409-414.

[3] P. Liu, Francis Lau, Michael J Lewis and Cho-li Wang. A New Asynchronous Parallel Evolutionary Algorithm for Function Optimization [A]. In: 7th International conference on parallel problem solving from nature [C]. Granada, Berlin: Springer. vol.2439, (2002),401-410.

[4] P. Liu, L.S. Kang, H.D Garis, and Y.P. Chen, An asynchronous parallel evolutionary algorithm (APEA) for solving complex non-linear real world optimization problems, Neural, Parallel & Scientific Computations, Vol.10, No.2, (2002),179-188.

[5] Y.S.Ong and A. J. Keane, Meta-Lamarckian Learning in Memetic Algorithms, http://ntu-cg.ntu.edu.sg/ysong/journal/IEEE_EC_Ysong2003.pdf, (2003).

[6] B.J. Zhen, Y.X. Li and M.C.Wu, Function Optimization Algorithm based on Cellular Automata, Computer Engneering, 29(19), (2003),66-67.

[7] R. Chelouah and P. Siarry, Tabu Search: Applied to Global Optimization [J], European Journal of Operational Research, 123(2),(2000),256-270.

[8] D. Cvijovic and J. Klinowski, Taboo Search: An Approach to the Multiple Minima Problem [J], Science, 267(5198), (1995), 664-666.

[9] Tahk. Min-Jea and Sun. Byung-Chan, Coevolutionary Augmented Lagrangian Methods for Constrained Optimization [J]. IEEE Transactions on Evolusionary Computation, 4(2), (2000), 114-124.

[10] X.F. Zou, L.S. Kang, Y.X. Li, "A dynamical evolutionary algorithm for constrained optimization problems," Proceedings of the 2002 Congress on Evolutionary Computation, vol. 1, (2002), 890-895.