Proceedings of the 2nd International Conference on Information Technology for Application (ICITA 2004)

A New Parallel Search Algorithm for Non-Linear Function Optimization

Hai Yan Lu, *IEEE, Member* and Guoli Zhang

Abstract-- **This paper proposes a new parallel search algorithm using evolutionary programming and quasi-simplex technique (EPQS). EPQS produces the offspring from three ways in parallel: 1) Using the Gaussian mutation, 2) Using the Cauchy mutation, and 3) Using the quasi-simplex techniques. The quasi-simplex technique uses the ideal of classical simplex technique and produces four prospective individuals by using the reflection, expansion and compression operations. EPQS selects the parents for the next generation from all the parents and offspring. EPQS takes the diversity of offerings into consideration by generating the offspring from as many as possible ways while it maintains a substantial convergence rate. Experimental studies on six typical benchmark functions have shown that the proposed algorithm is more effective than the competing algorithms.**

Index Terms– **Evolutionary programming, Gaussian mutation, Cauchy mutation, quasi-simplex, parallel search algorithm, non-linear programming**

I. INTRODUCTION

Evolutionary programming (EP) have been used to solve many combinatorial optimization problems and some practical problems. [1-3]. Nidul Sinha, R.Chakrabarti, and P.K.Chattopadhyay [3] and H. T. Yang, P. C. Yang, and C. L. Huang [4] used EP techniques successfully for economic load dispatch problems, which have non-convex cost curves where conventional gradient-based methods a re inapplicable.

Classical EP has some advantage over the Genetic Algorithm (GA) in such ways that EP relies primarily on mutation and selection and it does not need coding and decoding. However, the classical EP cannot compete with GA in terms of diversity of offspring in the evolution process and suffers a slow convergence rate in general. In order to get better performances, researchers have made great efforts to study EP techniques and the algorithm performances have increased evidently. Xin Yao and Yo ng Liu [5] proposed "fast EP" (FEP) algorithm which used a Cauchy mutation instead of Gaussian mutation as the primary search operator, and reached the conclusion that Cauchy mutation performances better than the Gaussian mutation because of its higher pro bability of making longer jumps and the capability to get the global optimum quickly. However, a large step size may not be beneficial at all if the current search point is already very close to the global optimum. Yong Liu, Xin Yao [6] investigated how different values of the step size parameter t impact on the performance of FEP and found that $t = 1$ was not the optimal value and the optimal t was problem-dependent. It has been proposed to use self-adaptation t as a way to deal with the search steps so that t can gradually evolve towards its near optimum although its initial value might not be optimal, but the paper didn't give specific method. Chang-Yong Lee and XinYao [7] discussed adaptive Lévy mutation, but restricted the parameter α to four discrete values. Thomas Philip Runarsson and Xin Yao [8] proposed continuous selection strategies, which are motivated by the need for achieving self-adaptation. Thorsten Schnier and Xin Yao [9] studied multiple representations problem in EP and showed empirically how multiple representations can benefit the search as a good search operator could. Xin Yao and Yong Liu [10] proposed a truly population-based approach that emphasizes population rather than the best individual, the method can often bring in several important benefits to evolvable hardware, including efficiency, accuracy, adaptiveness, and fault-tolerance. Guo Guanqi and Yu Shouyi [11] proposed evolutionary parallel local search (PELS) by using simplex, but computation cost is very high as using simplex for every generation.

Although EP algorithms have been developing greatly and found their applications widely, it is still very important and desirable to enhance their capability of global search and speed up their convergence rates to make them more suitable to solve large scale real world problems, such as the unit commitment problem in electric power system to facilitate the power market. This paper proposes a new parallel search algorithm using evolutionary programming and quasi-simplex technique (EPQS). EPQS produces the offspring from three ways: 1) Using the Gaussian mutation, 2) Using the Cauchy mutation, and 3) Using the quasi-simplex techniques in parallel. The quasi-simplex technique uses the ideal of classical simplex technique and produces four prospective individuals by using the reflection, expansion and compression operations. EPQS selects the parents for the next generation from all the parents and offspring. EPQS takes the diversity of offerings into consideration by generating the offspring from as many as possible ways while it maintains a substantial convergence rate. Experimental studies

This work was supported in part by the Faculty of Information Technology, University of Technology, Sydney, Australia, under the research seed fund 1272r.

Hai Yan Lu is with the University of Technology, Sydney, Australia. PO Box 123, Braodway NSW 2007, Australia (e-mail: helenlu@it.uts.edu.au)

Guoli Zhang is with North China Electric Power University, China. He is on study leave from his host university and works in UTS as a visiting scholar (e-mail: Zhangguoli@ncepu.edu.cn).

ICITA2004 ISBN 0-646-42313-4

on six typical benchmark functions have shown that the proposed algorithm is more effective than the competing algorithms.

The rest of this paper is organized as follows. Section 2 describes the global minimization problem considered in this paper and one of the most advanced EP methods, which is the improved fast evolutionary programming (IFEP) method proposed by Xin Yao and Yong Liu [5]. Section 3 describes the classical simplex technique. Section 4 proposes a new parallel search algorithm based on the combination of evolutionary algorithm and the idea of classical simplex search techniques. Section 5 presents the experimental verification of the proposed new algorithm by the comparison of the experimental results of the IFEP algorithm and the proposed new method. Section 6 concludes the paper and indicates the further study.

II. FUNCTION OPTIMIZATION BY THE IMPROVED FAST EVOLUTIONARY PROGRAMMING

Consider the global minimization problem described by Xin Yao and Yong Liu [5] for the purpose of development of new search algorithm. According to Yao and Liu, the problem can be formalized as a pair of real valued vectors (S, f), where $S \subseteq R^n$ is a bounded set on R^n and $f : S \to R$ is an n-dimensional real-valued function. Function f does not need to be continuous but it must be bounded. The problem is to find a point $x_{\min} \in S$ such that $f(x_{\min})$ is a global minimum on S. In other words, it is required to find an $x_{\min} \in S$ such that

$$\forall x \in S : f(x_{\min}) \le f(x) \tag{1}$$

and this paper only considers unconstrained function optimization.

According to Yao [5], the "improved fast EP (IFEP)" performs better or as good as the better of "fast EP (FEP)" and "classical EP (CEP) for most benchmark problems. Aiming at showing the effectiveness of our new search algorithm, the IFEP algorithm has been chosen to be the competing algorithm. For the ease of the comparison, the IFEP algorithm applied to function optimization in this study is described as follows:

Step 1: Generate the initial population of μ individuals, and set the counter $k = 1$. Each individual is taken as a pair of real-valued vectors, $(x_i, \eta_i), \forall i \in \{1, \cdots, \mu\}$, where x_i's are objective variables and η's are the standard deviations for Gaussian mutations.

Step 2: Evaluate the fitness score for each individual $(x_i, \eta_i), \forall i \in \{1, \cdots, \mu\}$ of the population based on the objective function $f(x_i)$.

Step 3: Each parent (x_i, η_i), $i = 1,2,\ldots,\mu$, creates a single offspring (x_i', η_i') by:

$$x_{i\_g}'(j) = x_i(j) + \eta_i(j) N_j(0,1) \tag{2}$$

$$x_{i\_c}'(j) = x_i(j) + \eta_i(j) \delta_j \tag{3}$$

$$\eta_i' = \eta_i(j) \exp(\tau' N(0,1) + \tau N_j(0,1)) \tag{4}$$

where the subscript j denotes the j-th component of the corresponding vectors, $x_{i\ g}'$ and x_{i_c}' represent the offspring generated from the Gaussian and the Cauchy mutations, the factors τ and τ' are commonly set to be $\left(\sqrt{2\sqrt{n}}\right)^{-1}$ and $\left(\sqrt{2n}\right)^{-1}$ respectively; $N(0,1)$ denotes a normally distributed one-dimensional random number with mean 0 and standard deviation 1. $N_j(0,1)$ denotes the random number that is generated for each value of j. δ_j is a Cauchy random variable with the scale parameter $t = 1$. The one-dimensional Cauchy density function centered at the origin is defined by:

$$f_t(x) = \frac{1}{\pi} \frac{t}{t^2 + x^2}, \quad -\infty < x < \infty, \tag{5}$$

The single offspring x_i' will be the better one of $x_{i\ g}'$ and $x_{i\ c}'$ in terms of the fitness score.

Step 4: Calculate the fitness of each offspring $(x_i', \eta_i'), \forall i \in \{1, \cdots, \mu\}$.

Step 5: Sort the union of parents (x_i, η_i) and offspring $(x_i', \eta_i'), \forall i \in \{1, \cdots, \mu\}$ as the ascending order of their fitness. (This is equivalent to the step 5 in [5] when q = μ).

Step 6: Select the top μ individuals out of the sorted union (x_i, η_i) and $(x_i', \eta_i'), \forall i \in \{1, \cdots, \mu\}$ as the parents in the next generation.

Step 7: Stop if the halting criterion is satisfied; otherwise, $k = k + 1$ and go to Step3.

III. CLASSICAL SIMPLEX METHOD

In this paper, we will propose a novel parallel search algorithm using the evolutionary techniques and simplex ideas. This section briefly describes the conventional simplex technique [12].

Simplex is a kind of direct search method, which is a widely accepted search technique. A simplex in an n-dimensional space is defined by a convex polyhedron consisting of $n+1$ vertices denoted as $\mathbf{x}^i$, i = 1,2,..., n+1. The iteration proceeds by predicting a point at which the function has a more desirable value, and then including this point in the simplex. Let f_i, i = 1,2,..., n+1, denote the values of the function at the points $\mathbf{x}^i$. The values are compared to determine

$$f(\mathbf{x}^H) = f_H = \max_i f_i, \quad i = 1,2,\ldots,n+1 \tag{6}$$

$$f(\mathbf{x}^G) = f_G = \max_i f_i, \quad i = 1,2,\ldots,n+1,\ i \neq H \tag{7}$$

and

$$f(\mathbf{x}^L) = f_L = \min_i f_i, \quad i = 1,2,\ldots,n+1 \tag{8}$$

where $\mathbf{x}^H$, $\mathbf{x}^G$, and $\mathbf{x}^L$ have the worst, the second worst and the b est function values, respectively. $\mathbf{x}^H$ is therefore the point to be replaced in the iteration process.

Of the points of the simplex, $\mathbf{x}^H$ now has the worst function value. It is therefore the point to be replaced.

To determine the new point, the centroid $\mathbf{x}^C$ of the remaining points is calculated by

$$\mathbf{x}^C = \left(\left(\sum_{i=1}^{n+1} \mathbf{x}^i \right) - \mathbf{x}^H \right) / n \tag{9}$$

The function is then evaluated at the reflection of $\mathbf{x}^H$ about this point,

$$\mathbf{x}^R = 2\mathbf{x}^C - \mathbf{x}^H \tag{10}$$

and this is referred as to the reflection operation.

If this point is not an improvement, i.e.

$$f(\mathbf{x}^R) \geq f_H \tag{11}$$

then too large a step has been taken in this direction and a contraction is made, the new point being defined by

$$\mathbf{x}^N = \mathbf{x}^H(1-\lambda) + \lambda\mathbf{x}^R \tag{12}$$

where $0 < \lambda < 1$, but $\lambda \neq 0.5$. This is referred as to the compression operation.

If $\mathbf{x}^R$ is an improvement, then there is a possibility that a larger step in this direction might be beneficial, i.e.

$$\mathbf{x}^E = \mathbf{x}^H(1-\lambda) + \lambda\mathbf{x}^R,\ \lambda > 1 \tag{13}$$

and this is referred as to the expansion operation.

Nelder and Mead recommended such an expansion only if

$$f(\mathbf{x}^R) < f(\mathbf{x}^L) \tag{14}$$

If the criterion is satisfied and the result is successful, i.e. $f(\mathbf{x}^E) < f(\mathbf{x}^R)$, then $\mathbf{x}^N := \mathbf{x}^E$. In any other circumstances, $\mathbf{x}^N := \mathbf{x}^R$. At this stage, a new point $\mathbf{x}^N$ has been defined. If $f(\mathbf{x}^N) < f(\mathbf{x}^G)$, then $\mathbf{x}^H := \mathbf{x}^N$, and the process is repeated.

If, however, $f(\mathbf{x}^N) \geq f(\mathbf{x}^G)$, the next probe would be along the same straight line. As this implies that the simplex is in some sense too large compared with the curvature of the function in this region, all the points $\mathbf{x}^i$ are replaced by

$$\mathbf{x}^i := (\mathbf{x}^L + \mathbf{x}^i)/2 \tag{15}$$

and the process then repeated. The (15) is referred as to the contraction operation.

The process will be continuing until the termination criterion is satisfied.

IV. A New Parallel Search Algorithm with Evolutionary and Quasi-Simple Technique

In order to increase the diversity of the offspring and provide a certain direction in the search process to speed up the convergence rate, we combine the evolutionary technique and the idea of classical simplex to formulate a new algorithm, which is called parallel search algorithm with evolutionary and quasi-simplex (EPQS). The main features of EPQS are as follows:

1) Taking the four potential good individuals generated based on the idea of the classical simplex technique as a part of the offspring in the evolution process. The crucial idea of the classical simplex technique is to predict the potential good points along the line that starts from $\mathbf{x}^H$ and towards $\mathbf{x}^C$ under the assumption that the best point is more likely along with this direction. The operations can be reflection, expansion, compression or contraction depending on the intermediate function values. EPQS takes this idea and generates four potential good individuals along the promising optimal direction by using operations of reflection, expansion and compression. These individuals are included in the offspring. Instead of using the full simplex approach which suffers a slow convergence rate and huge computational overhead in general, EPQS only generates four promising points in the offspring using the idea of simplex to get a sense of guide in the search process to search some possible global or local minimum points with the aim of increasing the convergence rate.

2) Making use of information of population rather than best individual information.

3) Using the rule of Gaussian mutation and Cauchy mutation in parallel to take the best world of the two mutations. Cauchy mutation intends to create large jumping steps and can enhance the capability of finding the global optimum while the Gaussian mutation intends to create small jumping steps and can enhance the capability of finding the local optimum. Gaussian mutation can increase the convergence rate when the search is close to the optimum while the Cauchy mutation enable to finding a better point, which is away from the current search area. Therefore both Gaussian and Cauchy mutation are used in the EPQS algorithm.

4) Following the nature evolutionary process. Every generation creates offspring whose size is bigger than the one for the parents. However, only the top μ individuals selected from all the parents and offspring in terms of their fitness will become the parents in the next generation.

The Quasi-simplex points in the Fig.1 can be obtained using the following algorithm:

Randomly select a set of $n+1$ individuals at a time from the population until the remaining individuals are less than $n+1$. Each set of n+1 individuals are considered as a simplex. For each simplex, four points will be generated using the reflection, expansion and compression (two points) operations. Quasi-simplex not only can speed up the convergence rate of local or global search, but also increase the useful population diversity, therefore it can reduce the probability of missing the global optimal in the search process.

The flow chart for the EPQS algorithm is depicted in Fig. 1.

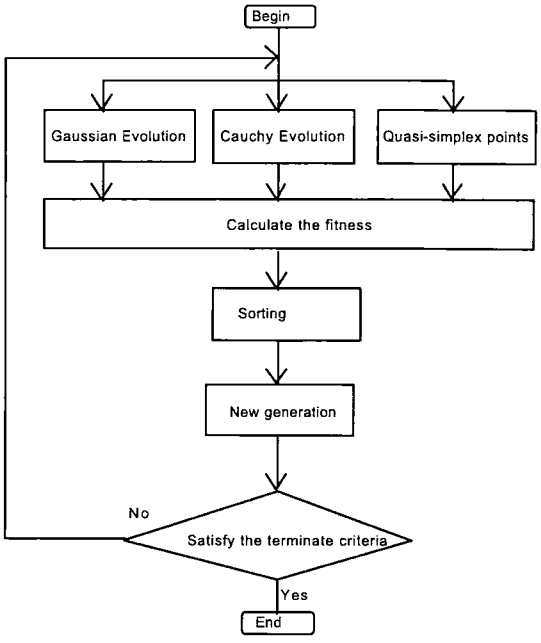

Fig.1 The flow chart of EPQS algorithm

V. EXPERIMENTAL VERIFICATION

A. *Test Set Up*

Aiming at testing the effectiveness of the proposed new algorithm in terms of the global search capability and the convergence rate, the IFEP algorithm described in section II and the new EPQS algorithms are implemented. Six functions were chosen from the set of 23 benchmark functions listed in [5] and re-numbered as f_1 *to* f_6, in which function f_1 and f_2 are typical unimodal functions; f_3 and f_4 are multimodal

Table 1 Definitions of test benchmark functions

No	Test function	Dimension	Domain	Minimum value f_{min}
1	$f_1 = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]^{30}$	0.0
2	$f_2 = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-100, 100]^{30}$	0.0
3	$f_3 = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i) + 20 + e$	30	$[-32, 32]^{30}$	0.0
4	$f_4 = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]^{30}$	0.0
5	$f_5 = -\sum_{i=1}^{5}[(x-a_i)(x-a_i)^T + c_i]^{-1}$	4	$[0,10]^4$	-10.0*
6	$f_6 = -\sum_{i=1}^{10}[(x-a_i)(x-a_i)^T + c_i]^{-1}$	4	$[0,10]^4$	-10.0*
Note: * given in [5][6][9], but we found they may not be the actual minimum values for the given functions.				

functions; and f_5 and f_6 are multimodal functions with only a few local minima. These functions are challenging to every search algorithm. The definitions of these six functions are depicted in Table 1. The coefficients a_{ij} and c_i in the functions f_5 and f_6 with i = 5 in f_5 and i = 10 in f_6 are shown in Table 2. Due to the time constraints, the dimension is set to be 10 instead of 30 in the functions f_1 to f_4.

Table 2 coefficients of functions f_5 and f_6

i	a_{ij}, j = 1,2,3,4				c_i
	j=1	j=2	j=3	j=4	
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

B. *Test Results*

The two algorithms use same parameters. The size of population is $\mu = 3(n+1)$. Table 3 tabulates the results obtained by running 10. The experimental results show the proposed algorithm EPQS is more effective than the competing algorithm on all the test benchmark functions in terms of both the mean function value and the standard derivation. From out experience in the computation, EPQS is also more stable and faster than the competing algorithm.

Another finding is that the last two functions, which have a few local minima, have smaller function values than the ones given in [5]. The minimal values for the functions f_5 and f_6 reached in the search processes are −10.1532 and −10.5364 using EPQA, respectively.

VI. CONCLUSIONS AND FURTHER STUDY

A new search algorithm EPQS algorithm for non-linear function optimization has been proposed based on evolutionary programming and quasi-simplex techniques. The new algorithm combines the evolutionary programming technique, such as Gaussian mutation and Cauchy mutation, and the idea of simplex method to produce the offspring from three ways in parallel. The parents for the next generation will be selected according to the fitness from all of the parents and offspring. This new algorithm has been compared with the IFEP algorithm under the same conditions over six typical benchmark functions. It has been shown that EPQS is more effective than the competing algorithm over all the six typical benchmark functions used in this study in terms of the mean function value and the standard derivation. It is also found that the given benchmark functions f_5 and f_6 have smaller function values than the ones given in the literature.

Further study would be to improve the proposed algorithm and develop applications using the algorithm.

ACKNOWLEDGMENT

The second author would like to thank the Faculty of Information Technology, University of Technology, Sydney (FIT in UTS), Australia, for supporting him to work in Sydney on this study.

REFERENCES

[1] Xin Yao, *An Overview of Evolutionary Computation,*

Table 3 Comparison between EPQS and IFEP on functions $f_1 - f_6$ listed in Table 1

Function	Number of Generation	EPQS		IFEP	
		Mean Best	Std Dev	Mean Best	Std Dev
f_1	8000	0.0514	0.3061	46.7067	194.892
f_2	5000	0.0300	0.1062	0.8987	2.1487
f_3	5000	0.9959	2.6254	4.2941	13.4172
f_4	8000	0.2131	0.3090	2.3895	5.2301
f_5	10000	-8.6591	9.4493	-6.3809	10.1987
f_6	10000	-9.7249	7.6983	-8.4168	10.2291

Note: All results have been averaged over 10 runs, where "Mean Best" indicates the mean best function values and "Std Dev" stands for the standard deviation.

Chinese J. Advanced Software Research, vol. 3, no. 1, pp.12-29, 1996.

[2] Xin Yao, ED., *Evolutionary Computation: Theory and Applications*, Singapore: world scientific, 1999.

[3] Nidul Sinha, R.Chakrabarti, and P.K.Chattopadhyay, *Evolutionary Programming Techniques for Economic Load Dispatch*, IEEE Trans. Evolutionary Computation, Vol. 7, No. 1, Feb. 2003.

[4] H. T. Yang, P. C. Yang, and C. L. Huang, *Evolutionary Programming Based Economic Dispatch For Units with Nonsmooth Fuel Cost Function*, IEEE Trans. Power Systems, Vol. 11, pp. 112-118, Feb. 1996.

[5] Xin Yao and Yong Liu, *Evolutionary Programming Made Fast, IEEE Trans. Evolutionary Computation*, Vol. 3, No. 2, pp. 82-102, July, 1999

[6] Yong Liu and Xin Yao, *How To Control Search Step Size in Fast Evolutionary Programming, Evolutionary Computation*, 2002. CEC '02. Proceedings of the 2002 Congress on , Vol.: 1, pp. 652-656, 12-17 May 2002.

[7] Chang Yong Lee and Xin Yao, *Evolutionary Algorithms with Adaptive Lévy Mutations, Evolutionary Computation*, 2001. Proceedings of the 2001 Congress on, Vol.: 1, pp. 568-575, 27-30 May 2001.

[8] Thomas Philip Runarsson and Xin Yao, *Continuous Selection and Self-Adaptive Evolution Strategies, Evolutionary Computation*, 2002. CEC '02. Proceedings of the 2002 Congress on, Vol.: 1, pp. 279-284, 12-17 May 2002

[9] Torsten Schnier and Xin Yao, *Using Multiple Representations in Evolutionary, Evolutionary Computation*, 2000. Proceedings of the 2000 Congress on , Vol.: 1, pp. 479-486, 16-19 July 2000

[10] Xin Yao and Yong Liu, *Getting Most out of Evolutionary Approaches*, Evolvable Hardware, 2002. Proceedings. NASA/DoD Conference on, pp. 8-14, 15-18 July 2002

[11] Guo Guanqi and Yu Shouyi, *Evolutionary Parallel Local Search for Function Optimization*, IEEE Trans. Systems, man and cybernetics-part B: cybernetics, pp.1-13, 2003

[12] Nelder, J. A. and Mead, R. *A simplex method for function minimization*, The computer Journal, 5, 1965

cations

ICITA 2004 Harbin, China http://www.icita.org

Second Inte[illegible]nal C
on Informa
& Applic

rbin,
8-11 January 2004
http://www.icita.org

Sponsored by:
[illegible], China
[illegible] University, China
[illegible]ession, Australia
[illegible] Chapter, Beijing, China
Saora Inc., Japan
papers published at the ICITA2004
will be EI Indexed.

Saora
Illuminating Interfaces

EEE