Asynchronous Distributed Parallel Gene Expression Programming based on Estimation of Distribution Algorithm

Xin Du[1,2],Lixin Ding[1],Liyuan Jia[3]
1 State Key Lab. of Software Engineering,Wuhan University,Wuhan 430072,China
2 Department of Information and engineering, Shijiazhuang University of Economics, Shijiazhuang, China,050031
3 Dept of computer ,Hunan City University,Yiyang,China,413000
Email: xindu79@126.com

Abstract

In order to reduce the computation time and improve the quality of solutions of Gene Expression Programming (GEP), a new asynchronous distributed parallel gene expression programming based on Estimation of Distribution Algorithm (EDA) is proposed. The idea of introducing EDA into GEP is to accelerate the convergence speed. Moreover, the improved GEP is implemented by an asynchronous distributed parallel method based on the island parallel model on a message passing interface (MPI) environment. Some experiments are done on distributed network connected by twenty computers. The best results of sequential and parallel algorithms are compared, speedup and performance influence of some important parallel control parameters to this parallel algorithm are discussed. The experimental results show that it may approach linear speedup and has better ability to find optimal solution and higher stability than sequential algorithm.

1. Introduction

GEP [1] was first proposed by Candida Ferreira which uses gene expression law of biological heredity for reference. Now, GEP has become a powerful tool of automatic programming. However GEP still has limitations. First, each mutation can lead to many different tree structures. Its searching is blind and some better structures cannot be retained. Second, like other evolutionary algorithms, it is still no guarantee for optimum solution (i.e. premature convergence) and has high computation time.

For improving the performance of GEP, the idea of EDA [2] is introduced into GEP which is to accelerate the convergence speed. Moreover, the idea of co-evolution of multi-population is introduced, which uses an asynchronous distributed parallel method based on the island parallel model on MPI environment to improve searching ability and reduce computation time. Some parallel experiments are done on parallel processing system.The best results of sequential and parallel algorithm are compared, speedup and performance influence of some important parallel control parameters are discussed. The experimental results show that it may approach linear speedup and has better ability to find optimal solution and higher stability than sequential algorithm.

The other parts of this article are organized as follows. Section2 introduces related work; Section3 explains asynchronous distributed parallel GEP based on EDA; Section4 does some experiments about linear speedup and influence of parallel parameters to parallel algorithm; Section5 presents some conclusions.

2. Related Work

GEP is a genotype/phenotype genetic algorithm. It uses fixed linear chromosomes composed of genes structurally organized in a head and a tail. This character assures that the performance of GEP is better than traditional GA [1, 4]. Usually, for improving the performance of GEP, other intelligent method is introduced into GEP. Bio-inspired Computing is a heurist intelligent computation method inspired from nature phenomenon or procedure. Bio-inspired Computing is introduced into GEP [5, 6]. For improving the learning ability of GEP ,learning mechanism is introduced into GEP [3, 7]. For keeping some good structures that may be eliminated from population, these individuals with good structure will get lower fitness due to improper parameters what they select. Two levels evolutionary method which uses GEP to optimize model structure and uses GA to optimize model parameters is used [8].

Inherent parallelism is regarded as one of the most im-

978-0-7695-3304-9/08 $25.00 © 2008 IEEE
DOI 10.1109/ICNC.2008.497

portant advantages of evolutionary algorithms. Some parallel genetic algorithms have been proposed. For example, Asynchronous Island Parallel GA is proposed [9] to improve performance of algorithm.

3. Asynchronous Distributed Parallel Gene Expression Programming based on EDA

3.1. Gene Estimation of Gene Expression Programming algorithm

Gene structure of traditional GEP has some deficiencies, take single gene chromosome for example, if the first symbol of head of gene is a terminator, then the chromosome has no value, because the expression corresponding to the gene is only a terminator. So we use improved gene structure [3] , the gene head in GEP is divided into a head and a body. This structure is benefit to introduce learning mechanism and solve more complex problem.

The head contains only functions, the body contains symbols representing both functions and terminals, whereas the length of the tail is a function of head ,body and the number of arguments of the function with more arguments n(also called max arity) and is evaluated by the equation: tail =(head+body)(n-1) + 1.

Homeotic genes have exactly the same kind of structure as conventional genes and are built using an identical process. Homeotic gene is used to connect conventional genes.

In order to improve the convergence speed of GEP, the idea of EDA is introduced into GEP, Gene Estimation of Gene Expression Programming(GEGEP) [3] algorithm is proposed. Considering time and space cost, Univariate EDA Model is used. The GEGEP algorithm is as follows:

PROCEDURE GEGEP algorithm
begin
  Initialize P(0);
  Evaluate every individual's fitness of P(0) and build probability table of estimation of distribution;
  t:=0;
  repeat
    Pm(t):=mutation{ P(t)} according to probability table of estimation of distribution;
    Pc(t):=crossover{ P(t)} ;
    Pt(t):=transpose{ Pc(t)} ;
    P(t+ 1):=selection{ Pt(t)} ;
    Renew probability table of estimation of distribution;
    t:= t+ 1;
  until termination condition;
  output solution of Pbest;
end

This main idea of GEGEP mutation is to carry on statistics to each operator of head and body of conventional genes and homeotic gene, extract its statistical probability, and then mutate according to its statistical probability as Table1. For example, if $k \in H$ and the *k*th position of G_i is operator ω , then mutates with probability p_k^ω , else mutates randomly. Here H contains head and body part. Take a single gene as an example: Let h = 2, b = 3, the number of statistical component (the head and body of gene) is $\mathrm{H} = \{1, 2, 3, 4, 5\}$, Function Set $\mathrm{F} = \{+, -, *, /\}$, $\omega \in F$ the fitness value of chromosome G_i is f_i, then $p_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$, so $\sum_{i=1}^{n} p_i = 1$; if the kth position of G_i is operator ω, then Z_{ij}^ω is 1, else Z_{ij}^ω is 0. $p_j^\omega = \sum_{i=1}^{n} Z_{ij}^\omega \cdot p_i, j \in H, \omega \in F$,Obviously $\sum_{i=1}^{n} p_i^\omega = 1$

Table 1. Probability of estimation of distribution

	j				
ω	1	2	3	4	5
+	$P1^+$	$P2^+$	$P3^+$	$P4^+$	$P5^+$
-	$P1^-$	$P2^-$	$P3^-$	$P4^-$	$P5^-$
$*$	$P1^*$	$P2^*$	$P3^*$	$P4^*$	$P5^*$
/	$P1^/$	$P2^/$	$P3^/$	$P4^/$	$P5^/$

3.2. Asynchronous Distributed Parallel Gene Expression Programming based on EDA

The asynchronous parallel algorithm is based on island-based parallel model [10] and uses MPI as parallel development platform. The migration topology is complete interconnection, each processor is connected with all other neighbor nodes. So migration individual can migrate to any other regions which is beneficial to population diversity. The algorithm is as below.

Step1: Produce P processes and every processor has a process. The ID number of each process is 0,1,...,P-1;

Step2: Process 0 reads datum, control parameters (including population size n, max evolutionary generation, migration interval etc), then sends these messages to other processes;

Step3: Each processor evolves independently (including initialization of population and probability table of estimation of distribution, selection, genetic operation and fitness computation of every individual);

Step4: Send m individuals to other processors for every s

generations (m duplicates still in original population). The migration individuals are the best individuals of population;

Step5: Check whether it has individual sent by other processors or not each generation. If it has, m worst individuals of present population are replaced by m individuals just received from other processors. If it has not, continue to do following operation;

Step6: When main processor reaches max generation, sends termination message to other P processors. The best model of main processor is the final best model and is output. Other P processors terminate when they receive termination message sent by main processor.

Characteristic of this algorithm:

1) The core of parallel programming is how to coordinate every node and assign assignments to each computation node evenly which is a main factor to influence performance of parallel programming. In our algorithm, population is evenly divided into some assignment blocks according to number of processors and each processor is responsible for computing a block so that it can balance assignment of each processor;

2) Although interval generations sending individual in each processor are fixed, the method of receiving individuals in each processor is that inspecting and judging whether it has a message or not, if it has, then receives, else waits. Thus this algorithm realizes complete asynchronous parallel;

3) Due to multi-population characteristic, the possibility of getting stuck in local optimums is less;

4) Each process communicates with other processes.This is benefit to exchange information between sub-population and enhances population diversity.Here lets m equal to 1 considering communication cost.

4. Experiments on parallel evolutionary algorithm and parallel parameters

We evaluate performance of parallel algorithm from three aspects: parallel speedup, average fitting error (AFE), average fitness value (AF). The quality of solution is evaluated by AFE and AF. AFE is given by the following formula, where m means numbers of records. $AFE = \sum_{i=1}^{m}\left[\sqrt{\sum_{j=1}^{n}(\bar{y}_i - y_i)^2}\right] \Big/ m$. The parallel speedup is shown by variable S_p. S_p is given by the formula $S_p = \frac{\bar{T}_1}{\bar{T}_p}$, Where $\bar{T}_1$ means average computation time of sequential algorithm, $\bar{T}_p$ means average computation time of parallel algorithm.

Table 2. Parameter setting

population size	800
head length of genes	1
body length of genes	10
head length of homeotic gene	6
body length of homeotic gene	20
numbers of genes	8
mutation rate	0.044
One-point recombination rate	0.3
Two-point recombination rate	0.3
gene recombination rate	0.1
IS transposition rate	0.1
RIS transposition rate	0.1
Gene transposition rate	0.1

Table 3. Parameter setting of speedup

S-1	Generation=1000,s=50
S-2	Generation=1000,s=100
S-3	Generation=1000,s=200
S-4	Generation=5000,s=100
S-5	Generation=5000,s=200
S-6	Generation=5000,s=500

4.1. Experimental setting

Our parallel environment is composed by twenty computers on 10Mbps Internet. Operation system is WINXP and parallel platform is MPI 1.2.4. Experimental data comes from the amount of gas emitted from coalfaces of literature [11]. For every experiment, we all run 100 times independently and then compute AFE, AF, average computation time and Sp. In order to compare parallel algorithm and sequential algorithm, we first run sequential algorithm for 100 times. The parameter setting is shown in table2.The numbers of processors are twenty, so sub-populations are twenty. The sub-population size is 40 because 800 divide 20 is 40.

4.2. Test of speedup

We use same computer configuration. In order to assure the stability of algorithm, we let sequential and parallel program run 100 times. Parameter setting is shown in table3. Variable s means migration interval, Variable Generation means evolutionary generation. From table4 we can see that six times parallel speedup can be got at least. We can obtain linear speedup at setting S-3. When we solve a same problem, asynchronous parallel algorithm can spend less time than sequential algorithm.

Table 4. Test result of speedup

	speedup
S-1	9.04
S-2	16.1
S-3	17.22
S-4	6.78
S-5	12.78
S-6	14.25

4.3. Comparison of sequential and parallel result

We compare the best result of sequential and parallel algorithm after 100 runs. The best model of parallel algorithm is: Y=Exp((Exp(((Sin(Ln(c))*(Exp(b)-Sqrt(c*f)))/(((d/Tan(f*e/b)) /Cos(f-Tan(e^2*d-e+a)))+f-Tan(d)))))/Cos(Sin(((((((d/Tan(f*e/b)) /Cos((Tan(Ln(c))-Ln(c))))-f+Tan(d))*Ln(c))/ (f-Tan(e^2*d-e+a))))))). Fitness is 987.5432, FE is 0.226017 and computation time is 114.2 seconds (about 2 minutes). Parameter setting: variable s is 100, variable generation is 5000, number of processes is 20. The fitting and prediction curve is shown in Figure1.

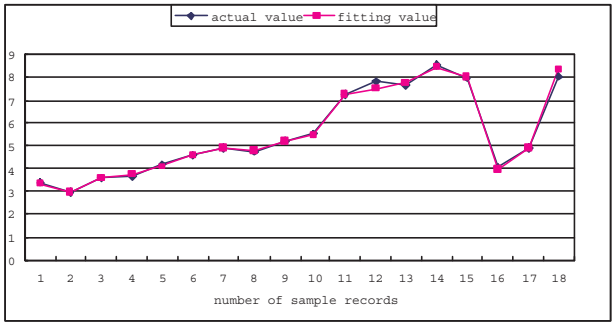

Figure 1. The curves of the best parallel evolutionary model

The best model of sequential algorithm is: Y=Ln(Exp((Exp(Cos((Cos((Cos(Sqrt(Exp(Sin(Tan(Sqrt(a+c)+Sqrt(f))))))-Expb))/Sqrt(Sqrt(Sqrt(Ln(((Expc-Tand)/Sqrt(Exp(Sin(Tan(Sqrt(a+c)+Sqrt(f)))))))))))+Ln((((Cos(Exp(Tan(d-b)))+Expb)-Sqrt(Exp(Sin(Tan(Sqrt(a+c) + Sqrt(f)))))) /Tan(LnLnd)))))), Fitness is 987.1156, FE is 0.218814, computation time is 1124 seconds(about 20 minutes). The fitting and prediction curve is shown in Figure2.

From the experimental results, we can see that computation time is reduced greatly. The model of parallel algorithm is almost as accurate as the model of sequential algorithm. From Figure1 and Figure2, we can see that fitting and prediction effect is very good.

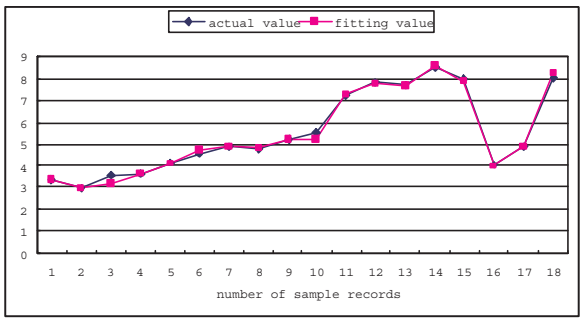

Figure 2. The curves of the best sequential evolutionary model

Table 5. Test result of speedup

S-1	s=50	S-3	s=200
S-2	s=100	S-4	s=500

4.4. The influence of parallel control parameter to quality of final solution

Migration interval is an important control parameter. This experiment tests the influence of migration interval. Parameter setting is shown in table5. Variable s means migration interval and variable generation means evolutionary generation. Here we let generation equal to 5000.

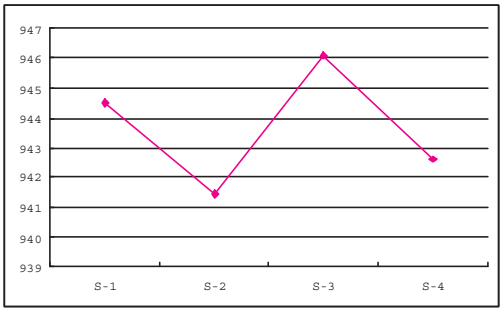

Figure 3. The influence of migration interval to average fitness

From Figure3 and Figure4, we can draw some conclusions:

(1) Longer migration interval is, less computation time is.

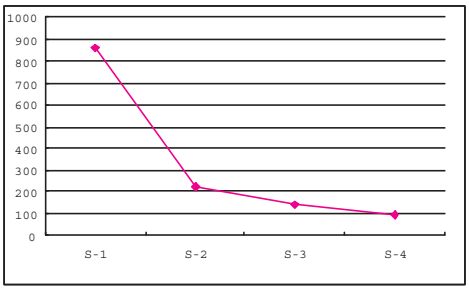

Figure 4. The influence of migration interval to average computation time

(2) Best average fitness value is got when migration interval is 200.

(3) Migration interval should not be set too frequent because that will spend a lot of communication time.

(4)Because average fitness of sequential algorithm is 940.4579, average fitness of parallel algorithm is better than that of sequential algorithm.

5. Conclusion

In this paper, EDA and multi-population strategies are introduced into GEP, an asynchronous distributed parallel GEP method based on the island parallel model is proposed, which improves searching ability and reduces computation time.

1) EDA is a good learning algorithm, which is benefit to improve convergence speed. Due to multi-population characteristic, the possibility of getting stuck in local optimum is less than single-population and individual migration lowers premature convergence and improves diversity.

2) The best results of sequential and parallel algorithm are compared, speedup and performance influence of some important parallel control parameters to this parallel algorithm are discussed. The experimental results show that it may approach linear speedup and has better ability to find optimal solution and higher stability than sequential algorithm.

Acknowledgements

This research was Supported by the Youthful Outstanding Talent Foundation in Hubei Province (Grant no. 2005ABB017),the Research Fund for the Doctoral Program of Higher Education of China(Grant no.20070486081), the High Science and Technology Research of Hebei Province of China under Grant (No.05213567, 06213562).

References

[1] Ferreira C, ”Gene Expression Programming: a New Adaptive Algorithm for Solving Problems”, Complex Systems, 2001,2 (13) ,pp. 87-129

[2] Larranaga P, Lozano J A, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, 2002

[3] Xin Du, Yueqiao Li et al”A New Algorithm of Automatic programming: GEGEP”The 6th International Conference on Simulated Evolution and Learning, Hefei, China, 2006,10,pp.292-301

[4] Ferreira C, ”Mutation , transposition , and recombination : An analysis of the evolutionary dynamics”, Proceedings of the 6th Joint Conference on Information Sciences, USA,2002, pp. 614-617

[5] Yixiao Zhong, Chang-jieTang et al, ”Improve KDD efficiency of Gene Expression Programming by Backtracking Strategy”, Journal of Sichuan University,2006,2

[6] Jing Peng,Chang-jieTang,et al, ”Evolutionary Algorithm Based on Overlapped Gene Expression”, ICNC 2005, LNCS 3612,pp.194-204

[7] Xin Li, Chi Zhou, Weimin Xiao etc.” Introducing Emergent Loose Modules into the Learning Process of a Linear Genetic Programming System”, Proceedings of the 5th International Conference on Machine Learning and Applications (ICMLA’06)

[8] Hongqing Cao, Jingxian Yu, Lishan Kang, ”An Evolutionary Approach for Modeling the Equivalent Circuit for Electrochemical Impedance Spectroscopy”, CEC’03, pp.1819-1825

[9] Hirosuke Horii et al, ”Asynchronous Island Parallel GA Using Multiform Subpopulations”, Second Asia-Pacific Conference on Simulated Evolution and Learning on simulated evolution and learning, Springer-Verlag, London, UK, 1998, pp.122-129

[10] CantuPaz E, ”A survey of parallel genetic algorithms. Calculateurs Paralleles”, Reseaux et Systems Repartis, 1998,10(2),pp.141-171

[11] Li Q, Cai ZH,Zhu L, Zhao SY, ”Application of Gene Expression Programming in Predicting the Amount of Gas Emitted from Coal Face”, Journal of Basic Science and Engineering, 2004,3(12),pp.49-54