A TRANSPUTER BASED PARALLEL COMPUTING ENVIRONMENT FOIR GENETIC ALGORITHMS

R.K.Ghosh,N.S.Arora and P.K.Kalra?

TRCS-94-226

SEPTEMBER-1994

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY:KANPUR
KANPUR-208016

A TRANSPUTER BASED PARALLEL COMPUTING ENVIRONMENT FOR GENETIC ALGORITHMS

R. K. Ghosh, N. S. Arora and P. K. Kalra[1]
Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur 208 016.
Email: *rkg@iitk.ernet.in*

Abstract

In this paper, we describe implementation of a transputer based parallel computing environment for experimenting on genetic algorithms (GAs) as problem solving paradigm for hard optimization problems. Both synchronous and asynchronous models of parallel computations with various organizations and processor topologies have been realized and their relative performance have been studied. These studies indicates performance of GAs improved in terms of speed and convergence in the parallel computation on transputers.

Introduction

Genetic Algorithms (GAs) are a set of sophisticated *search and improve* procedures based on the mechanics of natural genetics. Certain predesignated sequence of precise operators serve to guide the search which, otherwise, is practically blind. Three basic operators *viz.*, reproduction, crossover and mutation are used to solve the problem [4] at hand. These, simple and probabilistic, operators are decided statically before the commencement of search. Once the operators are determined the evolution of search progresses by repeated and probabilistic applications of these sequence of operators until an acceptable solution emerges. It is John H. Holland who pioneered GA as heuristic for seeking at least near optimal solutions to hard optimization problems with a reasonable computational effort [6] . The major advantages associated with GAs are its robustness, adaptability, parallelism and flexibility [4, 3]. GAs are free of problems typically associated with derivative methods. Successful applications of GAs have been shown in problems associated with mapping parallel tasks to parallel architectures [2], machine learning [4], controls [7], dependence test for program restructurer [9] and other hard optimizations [1].

[1]With Department of Electrical Engineering, IIT-Kanpur.

GAs are inherently parallel as search starts and progresses simultaneously along a number of trajectories in the search space. With the advent of parallel computer architecture, the development of efficient parallel GAs [8] promises exceptionally improved performance in terms of speed. In this paper a case study of implementation of GAs on transputer based parallel computing environment has been presented. The success of the implementation has demonstrated by applying GAs to a test bed consisting of De Jong functions [4] and the problem of system identification .

Implementation of Parallel GA

Parallel GAs (PGAs) reduce computational time by enabling parallel processings of GAs operators over the search space. Since large number of points are to be processed independently in a given domain, these can be performed in parallel by sub dividing the population on various processors. The probabilities of GA operators may vary for each processors. In addition, different types of crossover and mutation operators [3] can be used on different processor. Hence, speed, robustness and better convergence can be achieved by finetuning of these parameters appropriately through experimentation.

The implementation is done on a parallel computing platform consisting of a PC/AT-286 host and eight transputers. These transputers, T800 processors each with 1Mb RAM, are nnected physically as a linear array on a tram mother board. PARAS programming environment with concurrent runtime environment (CORE) supplied by Center for Development of Advance Computing (CDAC) has been used to implement parallel GAs. PARAS supports concurrency within a task through threads. The communication among various threads or tasks is performed via ports. The package runs in DOS environment and uses system call to execute the GA on transputer network. The following parallel computing paradigms have been implemented for parallelizing the GA operations *viz.* crossover, mutation, reproduction, and replacement:

- Linear diffusion model
- Cubic diffusion model.
- Shuffle diffusion model.

These models operate in synchronous and asynchronous modes. In the synchronous mode the communication is synchronized such that there is no generation gap, *i.e.*, member arriving at a particular transputer has been generated in the previous generation and not in the earlier. In the asynchronous mode communication and generation is done simultaneously, this leads to a generation lag. It has been observed that the quality of the basic GA is improved in the following ways:

- Preventing premature convergence.
- Introducing separate GA operators based on knowledge of search space local to individual processing elements.
- Controlled interactions among neighbouring sub-population.
- Speeding up the algorithm.

The package is menu driven and user friendly. The main menu has three options which are (i) GA menu, (ii) Plot menu, and (iii) Exit. The bare bone algorithm that executes on each transputer is of the following form:

```
generate initial sub-population
    while not evolution do {
        apply genetic operators to generate new sub-populations
        send and receive members from neighbouring sub-populations
    }
```

The GA Menu has options of executing the program on the transputers or DOS environment. It also provides the ability to switch between the plot menu and main menu. The transputer executable menu provides user with option to execute the various applications already compiled and in executable form. The plot menu can be used to observe convergence of GAs through plots drawn on screen. Optionally, the plots can be redirected to device files for printouts.

Test Bed for Parallel GA

This section describes the test bed for demonstrating the applicability of parallel GA environment. It consist of various De Jong [4] functions and a problem of system identification.

De Jong devised a family of functions to evaluate applicability of GAs as problem solving paradigms for optimizations. These functions include all kind of functions *viz.* convex, concave, continuous and discontinuous functions. Therefore, the evaluation of De Jong functions is considered as an objective assessment an computing environment for GA applications.

It is very difficult to know the model of the plant/process for a complex system. The standard technique [5] used to determine the model is called system identification. The problem can be stated as: "Given input signal and corresponding output response of the plant, determine the order of transfer function and the associated coefficients." Mathematically, the problem is framed as follows: suppose a general relationship between input-output is

$$\frac{Y(z)}{R(z)} = \frac{z^n + a_1 z^{n-1} + a_2 z^{n-2} + \cdots + a_n}{z^m + b_1 z^{m-1} + b_2 z^{m-2} + \cdots + b_m}$$

where n and m are numbers zeros and poles of two plant and a_i, b_i, indeterminates to be identified when input and output are known. For sake of simplicity the following transfer function have been selected here.

$$\frac{Y(z)}{R(z)} = \frac{z(z+0.5)}{z^2 + 1.5z + 0.7}$$

Results

The test functions are tried out on all the models and performance of different models are compared against corresponding sequential GAs. Due restriction in length it is not possible to incude all results and the associate plots. Only few representative cases have been cited here.

Linear diffusion model

The experimental runs with population size 30 for different De Jong functions show linear diffusion model performs much better than sequential GA in following respect

- fewer function evaluations required to obtain the optimum
- the fitness rise is faster as well as uniformly steeper
- excellent results were obtained in case of multimodal and discontinuous functions

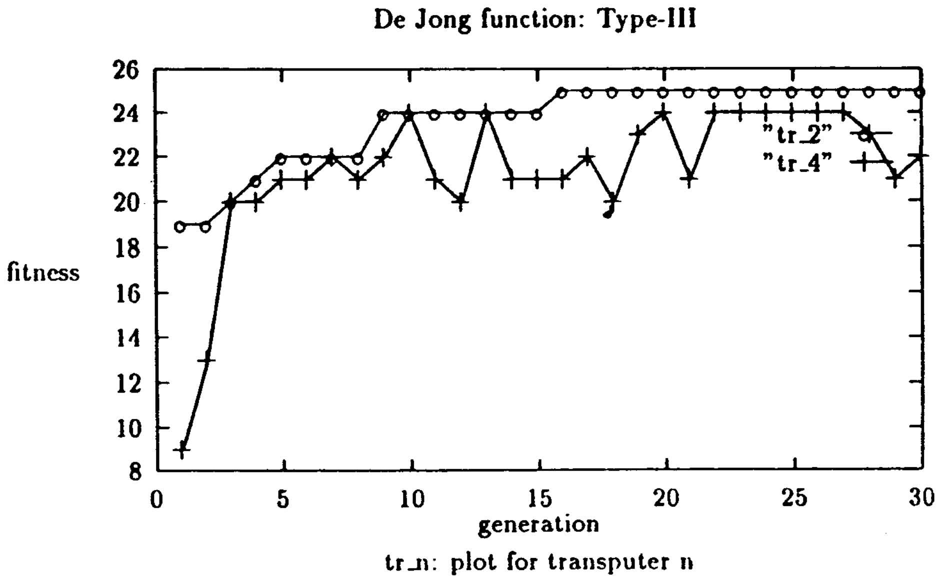


Figure 1: Plot for linear diffusion model

Interestingly, for De Jong's function of type III it was observed that the best result was obtained using 6 transputers, though the results generally improved as increasing number of transputers are employed. Essentially, speedup factor was not a linear function of number of processors as it would be expected. Figure 1 depicts the superimposed plots of the result of parallel computation of De Jong's function of type III on only two out of six different transputers. It provides indication that the large initial variation in genotype stabilizes in few generation due to exchange in sub-populations among processing elements.

Cubic diffusion model

The performance of this model is also much better than sequential GA. The convergence was faster and steeper. Again for multimodal and discontinuous functions this model proved to be applicable as opposed to ordinary GA. In certain cases the performance of the cubic model was better than linear diffusion model. But this is not a general pattern. The cubic diffusion model is found to be better than linear diffusion model in case of the third function in the list of De Jong functions. Since only two members were exchanged the effect of communication overheads on performance are considered to be not acceptable. The superimposed plots for the computation of De Jong functions of type II and type III are provided in Figure 2. Here, only results from two out of eight transputers are shown just give an idea about progress in computations.

Shuffle diffusion model

This model is based on perfect-shuffle interconnection scheme. Since exchange of sub-population takes place only in a predetermined manner the local variation in population structure induced by migration is limited. However, limited migrations ensure limited communication overheads. So, the treadoff in communication versus variation in local sub-population stru[illegible]ure can be balanced to some extent. For example, uni-directional links for shuffle and the bi-directional exchange links can be used in manner such that diffusion of genotype is gradual as would be expected in a natural setting.

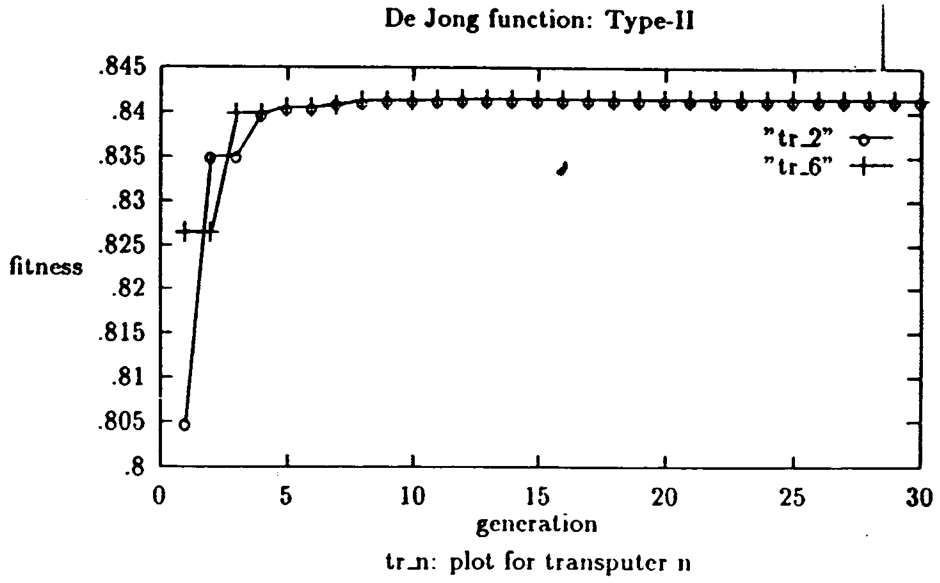
System identification problem

The most interesting observation was recorded for the problem of system identification. The single transputer run required to go through 800 generations of GA for an acceptable solution. But the same result was achieved just under 30 generations with 8 transputers on the cubic diffusion model and linear diffusion model. The other interesting point is cubic model produced the best result. For linear model the best result was recorded for 6 processors. The plots for the results obtained from cubic diffusion model are provided in Figure 3. Shuffle diffusion model provided results more or less comparable t› that obtained from cubic model. In fact, observations seem to confirm that controlled migration of sub-populations is most crucial to successful application PGA as a problem solving paradigm.

Enchancement of the Package

The implementation leaves scope to support open architectures. Currently, the work is in progress in this direction. The user may experiment with open architectures by specifying the following: (i) an operation mode (synchronous or asynchronous), and (ii) an arbitrary graph for processor interconnection.

The user interface of the PGA package have to be made window based if this package is to be targeted towards novice users. Accordingly, win-



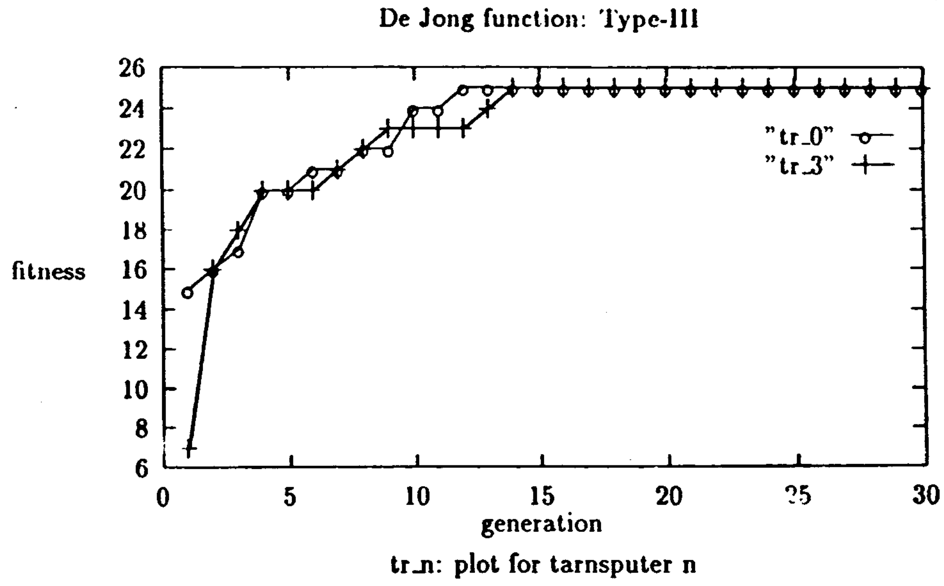


Figure 2: Plot for cubic diffusion model

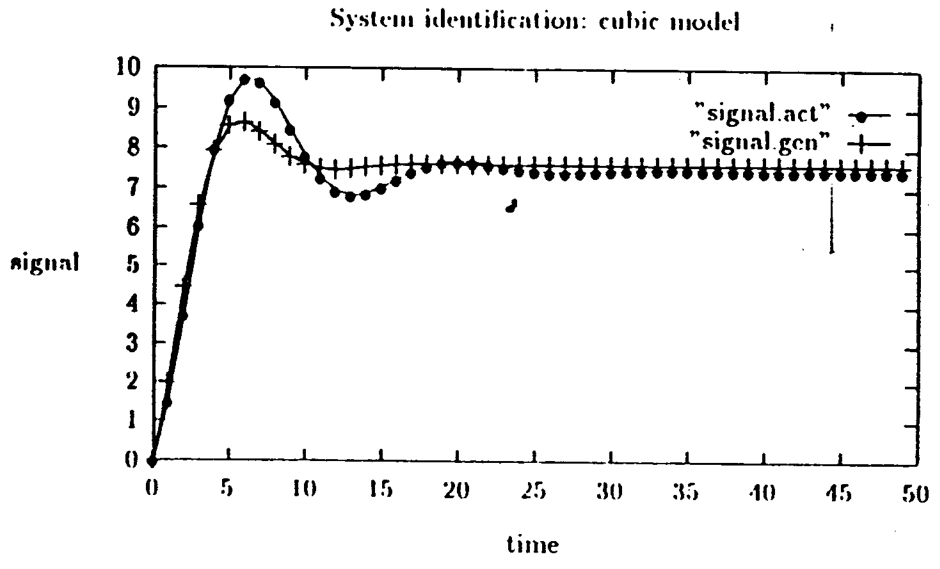


Figure 3: Plot for system identification

dow templates for different menu options with earlier version of all the programs have been designed and the coding windows for GA environment is in progress. It has been also planned to provide an on-line tutorial for GA windows.

On actual PGA side of the package, different non-binary coding schemes being tested and for possible implementation on the transputer based systems. Genetic Algorithms and its variations are being implemented for DC motor control. The future plans include provisions for incorporation of libraries for few application areas which include controls among others.

Acknowledgement

This research was supported under a grant CSIR/CSE/93024 by Council of Scientific & Industry Research (CSIR), Govt. of India. Authors wish to acknowledge C. Sudhakar for his help in integration of the manuscript.

References

[1] Chandramouli, A., Ghosh, R. K., Kalra, P. K., Srivastava S. C. and

Mishra, D. K., Genetic algorithms: Some applications to electric power systems, *Proceedings of 6th ISCA International Conference on Parallel and Distributed Computing & Systems*, pp. 405–410, Oct. 14-16, 1993, Louisville, Kentucky, USA.

[2] Chokalingam, T. and Arunkumar, S., A randomized heuristics for the mapping problem: The genetic approach, *Parallel Computing*, 18(1992), pp. 1157–1165.

[3] Davis, L., *Handbook of genetic algorithms*, Von Nostrand Reinhold, 1991.

[4] Goldberg, D. E., *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley, Reading, 1989.

[5] Gopal, M., *Digital control engineering*, Wiley Eastern, 1988.

[6] Holland, J. H., *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, 1975.

[7] Michalewicz, Z., Janikow, J. and Krawczyk, A modified genetic algorithms for optimal control problems, *Computers and Mathematics with Applications*, 23(1992), pp. 83-94.

[8] Ribeiro, J. L., Treleaven, P. C. and Alippi, C., Genetic-algorithm programming environments, *Computer Journal*, 27(1994), pp. 28–43.

[9] Sudheer, H. R., *GAtest: An exact dependence test for parallelization of loops*, M. Tech. thesis, 1992, Department of Computer Science & Engineering, IIT-Kanpur, INDIA.