Buf-PGA : A Butterfly Topology Parallel GA as A Function Optimizer

Fattaneh Taghiyareh* & Hiroshi Nagahashi**

Imaging Science and Eng. Laboratory , Tokyo Institute of Technology
4259 Nagatsuta, Midori-ku, Yokohama 226-8503 , Japan
Phone : +(81)-45-924-5478, Fax: +(81)-45-924-5479
*taheri@n.isl.titech.ac.jp
**longb@n.isl.titech.ac.jp

ABSTRACT: Parallel genetic algorithms are often very different from the "traditional" genetic algorithm proposed by Holland, especially with regards to population structure and selection mechanism. This paper presents a new parallel genetic algorithm as a function optimizer. In this paper, a proposed fine-grain parallel GA based on the butterfly topology, henceforth referred to as *Buf-PGA*, has been discussed and its performance is compared with a 2-dimensional grid fine-grain parallel GA. We have successfully applied *Buf-PGA* to a number of problems, including function optimization. De Jong standard functions are used as a benchmark problem and computer experiments have been utilized in combination with theoretical analysis. We have looked for optimal parameter setting and we have studied the relationship between system parameters and *Buf-PGA*'s performance using online and offline performance measurements.

KEYWORDS : parallel genetic algorithms, butterfly topology, fine-grain parallel GA, De Jong Functions, online, offline

Parallel GA Category	**Population**	**Evaluation**	**Selection & Crossover**	**Behavior** Compare to simple GA
Micro-grain Parallel GAs	a single Panmictic Population	Distributed among several processors	Global	No change but in speed
Fine-grain Parallel GAs	Large number of very small subpopulations	Each processor, independently	Selection and Crossover are restricted to local neighborhood	Explore different parts of the search space
Coarse-grain Parallel GAs	A few large subpopulations which exchange individuals occasionally	Each processor, independently	Selection and Crossover inside each subpopulation	Fundamental changes in the operation of the GA, Subpopulations evolve independently

Table 1: *A Comparison between different Parallel GAs*

1 Introduction

Genetic algorithms are randomized search algorithms based on natural genetic evolution. They combine survival of the fittest with a randomized, yet structured information exchange to form a search algorithm. In every generation, a new set of strings is created using bits and pieces of the fittest of the previous generation. They efficiently exploit historical information to predict on new search points with improved performance and often outperform conventional optimization methods when applied to different real-world problems, such as numerical function optimization, image processing, combinatorial optimization, machine learning, manufacturing system engineering and so on. A variety of schemes for parallelizing genetic algorithms have appeared in the literature. Parallel GAs may be categorized as follows:*micro-grain parallel GAs, fine-grain parallel GA*, and *coarse-grain parallel GAs*[Gordon(94), Lin(94)]. Fine-grain parallel GAs which are subject of this work, partition the population into a large number of very small subpopulations. The ideal case is to have just one individual for every processing element available as its subpopulation. This model is suited for massively parallel computers, however, it can be implemented on any other multiprocessor as well. There are some related research reported, using different topologies such as ladder-like, two dimensional grid, hypercube, and so on[Gorges(89)]. Although, some topologies like butterfly[Taghiyareh(94)] has attracted little attention[Taghiyareh(98)-a, Taghiyareh(98)-b]. The **butterfly topology** henceforth referred to as **butterfly**, is one the most efficient networks for parallel computation. Massive communications capability of the butterfly which is a significant attribute for fine-grain parallel GAs, provides a way to disseminate good solutions across the entire population. Therefore, this topology appears to be a suitable choice for the topology of a fine-grain parallel GA.
In this paper, a proposed fine-grain parallel GA based on the butterfly topology, henceforth referred to as *Buf-PGA*, has been discussed and its performance is compared with a 2-dimensional grid fine-grain parallel GA. Our implementation deals with several subpopulations distributed in the network, which leads to overcoming the problem of premature convergence. Different regions of the search space are being evaluated at the same time and variability has been maintained for a longer period. Taking a hint from nature, we have local selection and local fitness-distribution. Natural selection is a local phenomenon taking place in an individual's local environment, however, individuals are not restricted to their local environment all the time. Following this fact, our proposed neighborhood is not fixed and can be expanded with a not-so-high probability.
We have successfully applied *Buf-PGA* to a number of problems, including function optimization. De Jong standard functions are used as a benchmark problem and computer experiments have been utilized in combination with theoretical analysis.
The major purpose of this work is to study the effects of some system parameters like *Range, Population-size,* and *Dimension* on the performance of the system. These parameters are described in following sections. We have looked for optimal parameter setting and we have studied the relationship between system parameters and *Buf-PGA*'s performance.

2 Background of work

2.1 Parallel Genetic Algorithms

A variety of schemes for parallelizing genetic algorithms have appeared in the literature. Some of the parallel implementations are very different from the "traditional" genetic algorithm proposed by Holland, especially with regards to population structure and selection mechanism. It is reasonable to assume that structural changes in a genetic algorithm will affect it's ability to solve problems. If the changes result in faster problem-solving, then the parallel models might be preferable even in the absence of parallel hardware. Parallel GAs promise substantial gains in performance of GAs. In addition to decreasing processing time, they explore the search space better. Unlike sequential GAs which pay a high computational cost for maintaining subpopulations based on similarity comparisons, parallel GAs (expect one model which is called micro-grain parallel GAs and explained later) maintain multiple, separate subpopulations which may be allowed to evolve independently. This allows each subpopulation to explore different parts of the search space, each maintaining its own high-fitness individuals and controlling how mixing occurs with other subpopulations, if at all.
There are three main categories of parallel GAs :**(1) micro-grain parallel GAs, (2) fine-grain parallel GAs, and (3) coarse-grain parallel GAs** .
In a micro-grain GA there is a single panmictic population (just as in a simple GA), but the evaluation of fitness is distributed among several processors[Huang(91)]. Fine-grain parallel GAs are suited for massively parallel computers and consist of spatially-structured population. Selection and mating are restricted to a small neighborhood, but neighborhood's overlap permits some interaction among all the individuals. Due to the overlapped neighborhoods, good solutions may disseminate across the entire population[Manderick(89)]. Coarse-grain parallel GAs consist on several subpopulations which exchange individuals occasionally. This exchange of individuals is called migration and it is controlled by several parameters. This model introduces fundamental changes in the operation of the GA and has a different behavior than simple GA[Gordon(94), Muhlenbein(91)].
It is important to emphasize that while the micro-grain parallelization method does not effect the behavior of the algorithm, the last two methods change the way the GA works. For example, in micro-grain parallel GAs, selection takes into account all the population, but in the other two parallel GAs, selection only considers a subset of individuals. Also, in the micro-grain any two individuals in the population can mate, but in the other methods mating is restricted to a subset of individuals. Table 1 summarizes similarities and differences between these three categories in comparison with *traditional* GA.
In conclusion, the answer to the question which parallel GA is the best is problem dependent. There is no universally best algorithm which can achieve the best result for all problems.

2.2 Parallel Topologies

There are two parallel topologies which are used in this work: *Butterfly* and *Mesh* or *two-dimensional grid* .

- **BUTTERFLY** The N-node butterfly can simulate any $O(N)$-node array, binary tree, or mesh of trees with only a small constant factor slowdown[Koch(97), Bhatt(88)].

  Definition 1 *The r-dimensional butterfly has* $(r+1)2^r$ *nodes and* $r*2^{r+1}$ *edges.The nodes correspond to pairs* $< w, i >$ *where* i *is the* level *or* dimension *of the node* $(0 \leq i \leq r)$ *and* w *is an* r*-bit binary number that denotes the row of the node. Two nodes* $< w, i >$ *and* $< w', i' >$ *are linked by an edge if and only if* $i' = i + 1$ *and either*

  1. w *and* w' *are identical*

  or

  2. w *and* w' *differ in exactly the* i'*th bit.*

  Butterfly is a systolic network which has a simple recursive structure and several nice properties which make it appropriate for many of parallel algorithms[Taghiyareh(94)].

- **MESH** We have compared the obtained results from our proposed algorithm with those from mesh-based parallel GA, and here is a very brief definition of this topology.

  Definition 2 *A n-processor mesh is a* $\sqrt{n} * \sqrt{n}$ *matrix of processors, each with degree 4 except for outermost rows and columns.*

  Mesh or 2-dimensional grid is a widely used network for fine-grain parallel GA because in many massively parallel computers the processing elements are connected using this topology.

$$f_1(x_i) = \sum_{i=1}^{3} x_i^2 \qquad -5.12 \leq x_i \leq 5.12$$

$$f_2(x_i) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 \qquad -2.048 \leq x_i \leq 2.048$$

$$f_3(x_i) = \sum_{i=1}^{5} integer(x_i) \qquad -5.12 \leq x_i \leq 5.12$$

$$f_4(x_i) = \sum_{i=1}^{30} ix_i^4 + \text{ Gauss(0,1)} \qquad -1.28 \leq x_i \leq 1.28$$

$$f_5(x_i) = 0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6} \qquad -65.536 \leq x_i \leq 65.536$$

Table 2: *De Jong Functions: Formula*

Functions	$\mathbf{F_1}$	$\mathbf{F_2}$	$\mathbf{F_3}$	$\mathbf{F_4}$	$\mathbf{F_5}$
Variables NO	3	2	5	30	2
Chromosome Length	30	24	50	240	34
Global Optimum	78.6	3905.93	55	1248.2	500

Table 3: *Properties of the De Jong Functions*

3 Test Functions and Setup of the Experiments

3.1 De Jong Functions

De Jong suite contains five test functions[Goldberg(89), Chapter 4]. f_1 is a unimodal function known to be easy for genetic algorithms. f_2 is a harder multimodal function. f_3 is a discontinuous "step ladder". f_4 involves a large solution space,i.e. 2^{240} plus Gaussian noise. f_5 is characterized by the presence of several local minima. De Jong suite are illustrated in Table 2. We applied binary representation for the chromosome and the most important properties of the test functions and their representation are summarized in Table 3.

3.2 De Jong Measurements

In order to evaluate the algorithm, some performance measures are required. We used the standard online and offline as performance measures in our experiments.

$$online(T) = \frac{1}{T}\sum_{1}^{T} mean(t) \tag{1}$$

$$offline(T) = \frac{1}{T}\sum_{1}^{T} best(t) \tag{2}$$

where $mean(t)$ is the mean fitness of the subpopulation at generation t, and, $best(t)$ is the fitness of the fittest individual of the subpopulation at generation t. In other words, the online performance is an average of mean-fitness up to the current generation (including this generation) and the offline is a running average of the best performance values to a particular generation. De Jong devised these two standard performance measures for the first time[Goldberg(89), Chapter 4]. Offline performance to gauge convergence and online to gauge ongoing performance. Since in a hardware implementation of the proposed algorithm, each generation would be evaluated in parallel, above mentioned measures slightly differ with standard performance measures introduced by De Jong.

In our experiment, the measures were computed for each processor of the network, separately. Since our implementation of the algorithm was a software simulation, we were able to compute these performance measures

- $Buf\text{-}PGA(D, R, S, P_c, P_m)$ denotes a particular implementation of butterfly-based fine-grain parallel GA.
- $Mesh\text{-}PGA(G, R, S, P_c, P_m)$ denotes a particular implementation of mesh-based fine-grain parallel GA.

D : Dimension
G^i: $1 \longrightarrow a\ 2*2$ Grid of nodes
$2 \longrightarrow a\ 4*4$ Grid of nodes
$3 \longrightarrow a\ 6*6$ Grid of nodes
R : Range
S : Size of subpopulation
P_c: Probability of crossover
P_m: Probability of mutation

[i] $G = 2$ for $FGA\text{-}M$ denotes a $4*4$ grid of nodes because it has 16 processors and it is the most proper size to be in corresponding with 2-dimensional butterfly. Consequently a $6*6$ grid of nodes is selected as a suitable correspondent for 3-dimensional butterfly.

Table 4: Notations

for entire population, too. It means first we computed online and offline performance for each subpopulation at every generation and after that total-online and total-offline were computed. The same formula was used however $mean(t)$ denotes the mean fitness of whole of population (all the subpopulations) and $best(t)$ represents the fittest individual at generation t between all the processors.

3.3 Buf-PGA

A panmictic population is divided into many subpopulations distributed among processors of butterfly. However there is no limitation in the size of subpopulation, it is desired to have a few individuals in each processor. Local subpopulations are initiated inside the node randomly. Each individual which is a solution to a given problem, is represented by a string of values called chromosome. Fitness of all the individuals in the population are evaluated in parallel by each processor locally and there is no need to any communication till this step of the algorithm.
Mimicking nature, selection is being done in the neighborhood of individuals which is not fixed and can be expanded with a not-so-high probability. The smallest neighborhood includes the node itself and its first-degree neighbors, which are connected via one edge to this node: this makes four neighbors for most of the processors. The exception is only on the first and the last level of the butterfly, where each node has only two first-degree neighbors. Giving different values to "*Range*" parameter causes changes in the neighborhood. For example for *Range*=1 neighborhood is restricted to the first-degree neighbors while *Range*=2 includes the second-degree neighbors, too. At each stage of the algorithm, every node of the butterfly gathers all the individuals from its neighborhood with its own subpopulation into a pool called *selection pool* and then selects n individuals randomly using a roulette wheel method. n individuals are selected from a pool including $(1+P)*n$ individuals. n denotes the size of each subpopulation and P stands for number of processors in the predetermined neighborhood. After conducting the *selection* step, each processor includes a list of selected individuals. two individuals are taken from the list with a given probability of *crossover-rate,* P_c, and are cross-bred to form offspring. With an exhaustive experiments the **uniform crossover** was chosen as our crossover technique. In *uniform crossover* method, each gene in the offspring is created by copying the corresponding gene from one or the other parent, chosen according to a randomly generated crossover mask. Uniform crossover has the advantage that the ordering of genes is entirely irrelevant.
The third GA operator, *mutation* operates on new population obtained by the above step and mutates each bit of chromosome with the probability of *mutation-rate,* P_m.
In this way, the new subpopulation is formed which should be evaluated. As a final step, the old subpopulation is substituted with the new subpopulation in each processor.

4 Experimental Results

The *Butterfly*-topology fine-grained PGA, $Buf\text{-}PGA$, has been simulated and its performance is compared with

Architecture	Mesh & Butterfly
Dimension	2-dimensional & 3-dimensional
Range of Neighborhood	1 & 2
Sub Population Size	set {1,2,5,10}
Crossover rate	$0.1, \ldots, 1.0$, $step:$ 0.05
Mutation Rate	$0.0001, \ldots, 0.1$
Selection Mechanism	Roulette Wheel
Crossover Mechanism	Uniform Crossover
Generations	100
Average over	20 runs

Table 5: *Buf-PGA and Mesh-PGA Parameters*

the *Mesh*-topology fine-grained PGA, *Mesh-PGA*. The online and offline performance were evaluated on the test functions with both *Buf-PGA* and *Mesh-PGA*. A summary of various notations used in this section are given in the Table 4. Each *Buf-PGA* and *Mesh-PGA* was tested on the test functions over 100 generations for a number of runs. [De Jong(75)] suggested a minimum of 5 runs, in order to deal with the variance in the performance of the individual runs. Therefore we averaged the performance measures over 20 runs. In addition we studied the relationship of *Range* and *Sub-population-Size* with the performance. Set-up of *Buf-PGA* and *Mesh-PGA* was the same for all of the experiments and is exhibited in Table5.
We looked at both the online and offline performance of *Buf-PGA* for a range of crossover and mutation rates. In our experiments, seven different crossover rates were used varying from 0.1 to 1.0 in increments of 0.05. The mutation rates were taken from the set { 0.0000, 0.0001, 0.0010, 0.0050, 0.0100, 0.0500, 0.1000 }.
Finding proper balance between exploration and exploitation is determinant for the performance of genetic algorithms. Too much exploitation may result in premature convergence. Especially for the offline performance it is important that the search space is explored sufficiently. Since *Buf-PGA* had a better offline performance on the test functions than *Mesh-PGA* , we hypothesized that *Buf-PGA* explores the search space more fully than *Mesh-PGA*.
Our findings can be summarized in a number of observations. A first observation concerns the relation between the mutation rate and the performance. In most cases, the best P_m at offline performance is comparatively higher than correspondent P_m at online performance. As an example, f_5 at online requires P_m ranging from $0.000 \sim 0.0010$ while the best P_m at offline performance seems to lie in the range of $0.0500 \sim 0.1000$. Other functions have the same situation at P_m, however the difference in the range of the best P_m is not as high as f_5. however f_5 represents the highest difference between offline's and online's P_m.
Another observation is in the relation between size of the sub-population and mutation rate. When $S = 5$ *or* 10, there is a tendency towards lower P_m. It may be due to the fact that with a larger subpopulation in each processor, there is enough variety in the population and therefore there is no need to apply higher P_m in order to explore the search space. It has to be noticed that when the function is a multimodal one it is important that the search space be explored sufficiently. Our results are consistent with this rule and average P_m required for f_5, a multimodal function, is higher than the same value for other simpler functions.
Another observation lies in the field of *Range*. As may have been expected when R varied from 1 to 2 (or vice versa), there was no considerable change in the best values found for P_m and P_c. The most likely reason for this behavior of the system is the fact that R is an independent variable which does not effect D and S. Increasing R only leads to an increase in the size of the selection pool which does not affect any other subsequent changes in the system. Table 6 illustrates some of representative results.

5 Conclusion and Future Work

In this paper, we have discussed a proposed fine-grain parallel GA based on the butterfly topology. One of its most important characteristics is varying neighborhood around each node which is the area for conducting local selection and crossover. Consequently, there is less selection pressure and a tendency towards more exploration of the search space. The algorithm can be implemented on massively butterfly-based parallel computers and opens perspective for experimenting with very large populations. We have looked for optimal parameter setting

(a)

Performance Measurement	Best Parameters for Buf-PGA					Performance Value
	Dimension	Range	Sub-population	P_m	P_c	
Online	3	2	10	0.0001	0.7	49.14
Offline	3	2	10	0.0001	0.8	54.22

(b)

Performance Measure	Best Parameters for Mesh-PGA					Performance Value
	Dimension	Range	Sub-population	P_m	P_c	
Online	3	2	10	0.0010	0.35	40.89
Offline	3	2	10	0.0250	0.7	42.85

(c)

Performance Measure	Buf-PGA	Mesh-PGA	Improvement
Online	49.14	40.89	20.18%
Offline	54.22	42.85	26.35%

Table 6: A representative result for comparing $Buf\text{-}PGA(3,2,10,P_c,P_m)$ and $Mesh\text{-}PGA(3,2,10,P_c,P_m)$

and found high quality solutions with a few generations. We have tested our algorithm by De Jong test funcions which are supposed to be a well-known benchmark for genetic algorithms and have compared the results with a mesh which is extensively used for parallel GAs. Based on the results, there is a good possibility that our proposed algorithm may outperform mesh-based parallel GAs, in a majority of problems. Moreover, findings lend support to the assumption that the butterfly is a promising network for implementing fine-grain parallel GAs. They suggest that it may be beneficial to apply butterfly-based fine-grain parallel GA to some real-world problems.
Currently, we are doing further research to obtain a better understanding of the behavior of the $Buf\text{-}PGA$. Furthermore we are trying to identify some guidelines for selecting the probability of crossover and mutation. It might be beneficial if we can combine $Buf\text{-}PGA$ with a fuzzy GA to estimate the best values for the crossover and mutation probabilities. In addition, we have started an experiment about some image processing applications using $Buf\text{-}PGA$ to recover the epipolar geometry of a moving object.

REFERENCES

[Bhatt(88)] Sandeep N. Bhatt, Fan R. K. Chung, Jia-Wei Hong, F. Thomson Leighton, and Arnold L. Rosenberg , May 1988 *Optimal simulations by butterfly networks*, In Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, pages 192-204, Chicago, Illinois.

[De Jong(75)] De Jong, K.A., 1975, *An Analysis of the behavior of a class of genetic adaptive systems* , Doctoral Dissertation, Univ. of Michigan, Ann Arbor, MI.

[Huang(91)] Fogarty, T. C., & Huang, R. , 1991, *Implementation the GA on Transputer Based Parallel Processing Systems.* PPSN-91, pp-145-149.

[Goldberg(89)] Goldberg, D. E.,1989, *Genetic Algorithms in Search, Optimization & Machine Learning,* Addison-Wesley Publishing Company.

[Gordon(94)] V. S. Gordon, D. Whitley,1994,*Serial and Parallel Genetic Algorithms as Function Optimizers* . Proceeding of the First IEEE Conference on Evolutionary Computation, Vol. 1, pp.428-432.

[Gorges(89)] M. Gorges-Schleuter,1989, *ASPARAGOS: A population genetics approach to genetic algorithms*, Evolution and Optimization 1989(pp. 86-94). Berlin: Akademie-Verlag.

[Koch(97)] Richard R. Koch, F. T. Leighton, Bruce M. Maggs, Satish B. Rao, Arnold L. Rosenberg, and Eric J. Schwabe., 1997, , *Work-preserving emulations of fixed-connection networks*, Journal of the ACM, 44(1):104-147, January 1997.

[Lin(94)] S.C. Lin, W. Punch, & Goodman, E., 1994, *Coarse-grain Parallel GAs : Categorization and New Approach.* Sixth IEEE Symposium on Parallel and Distributed Processing, IEEE Computer Society Press.

[Manderick(89)] Manderick, B. et. al, 1989, *Fine Grained Parallel Genetic Algorithms*, Proc. of the 3rd Int. Conf. on GAs, pp428-433.

[Muhlenbein(91)] Muhlenbein, H., Schomisch, M., & Born, J., 1991, *The Parallel Genetic Algorithm as Function Optimizer.* Proceeding of the Forth International Conference on GAs., Morgan Kaufmann.

[Taghiyareh(98)-a] F. Taghiyareh, H. Nagahashi, 1998, , *A Study about Parallel GA based on Butterfly Topology*, In Proceeding of 2nd International Symposium on Intelligent Manufacturing Systems, pages 245-255, Sakarya Univ. Turkey, August 1998.

[Taghiyareh(98)-b] F.Taghiyareh, H.Nagahashi, 1998, "BUFPGA: A Parallel Genetic Algorithm based on Butterfly Topology" , Proc. of the IEICE General Conference, D-8-19, Apr. 1998.

[Taghiyareh(94)] F. Taghiyareh, M. Ghodsi, 1994, , *Simulation of Running Parallel Algorithms on The Butterfly Network*, Master dissertation, in *Persian*, Sharif Univ. Tehran.