An overview of GA and PGA

R. K. Nayak
School of Computer Engg. KIIT University

B. S. P. Mishra, PhD
School of Computer Engg. KIIT University

Jnyanaranjan Mohanty, PhD
School of Computer Application, KIIT University

ABSTRACT
Genetic algorithms have been proven to be both an efficient and effective means of solving certain types of search and optimization problems. Genetic algorithms have been applied with positive results in many areas including scheduling problems, neural networking, face recognition and other NP-complete problems. The idea behind GA´s is to extract optimization strategies nature uses successfully - known as *Darwinian Evolution* - and transform them for application in mathematical optimization theory to find the global optimum in a defined *phase space*. Another popular way to improve genetic algorithms is to run them in parallel, some parallel genetic algorithms have performed very well compared to the standard non-parallel genetic algorithm. Parallel genetic algorithms focus their efforts at simulating multiple species and include not only the standard operations for crossover and mutation but also operations for migration between different populations.

Genetic algorithm (GA) which is a meta-heuristic algorithm has been successfully applied to solve the scheduling problem. The fitness evaluation is the most time consuming GA operation for the CPU time, which affects the GA performance. This paper proposes and implements a synchronous master-slave parallelization where the fitness evaluated in parallel. The rest of paper organized as follow: genetic algorithm, parallel genetic algorithm, proposed algorithm, theoretical analysis, practical analysis, and conclusion.

Keywords
Genetic Algorithm, Parallel Generic Algorithm, Dual Species, Genetic Algorithm, Search Algorithm, Path finding, GA, PGA, DSGA

1. INTRODUCTION
Genetic algorithms (GA) were first introduced by John Holland in the 1970s (Holland 1975) as a result of investigations into the possibility of computer programs undergoing evolution in the Darwinian sense [2]. GA are part of a broader soft computing paradigm known as evolutionary computation. They attempt to arrive at optimal solutions through a process similar to biological evolution [3]. This involves following the principles of survival of the fittest, and crossbreeding and mutation to generate better solutions from a pool of existing solutions [4]. Genetic algorithms have been found to be capable of finding solutions for a wide variety of problems for which no acceptable algorithmic solutions exist. The GA methodology is particularly suited for optimization, a problem solving technique in which one or more very good solutions are searched for in a solution space consisting of a large number of possible solutions [5]. GA reduce the search space by continually evaluating the current generation of candidate solutions, discarding the ones ranked as poor, and producing a new generation through crossbreeding and mutating those ranked as good [7]. The ranking of candidate solutions is done using some pre-determined measure of goodness or fitness.

2. DETAILS OF GENETIC ALGORITHM
A genetic Algorithm is an iterative procedure maintaining a population of structures that are candidate solutions to specific domain challenges. During each temporal increment (called a generation), the structures in the current population are rated for their effectiveness as domain solutions, and on the basis of these evaluations, a new population of candidate solutions is formed using specific genetic operators such as reproduction, crossover, and mutation. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure. While randomized, genetic algorithms are no simple random walk. They efficiently exploit historical information to speculate on new search points with expected improved performance.

2.1 Working of genetic algorithm
In the GA, each chromosomes are subjected to an iterative evolutionary process until a minimum the termination condition is met. The evolutionary process is carried out as in ordinary GA using genetic operators (crossover, mutation) and selection operations on chromosomes for reproduction. At every generation stage, parents are selected for mating and reproduction. A problem specific crossover operator that ensures solutions generated through genetic evolution are all feasible is also proposed. Hence both checking of the constraints and repair mechanism can be avoided, thus resulting in increased efficiency. Mutation is used to keep diversity in the population. Working of GA is shown in the following algorithm.

GENETIC ALGORITHM

```
Start
Read problem instance data
Set GA parameters
Initialize population
Initialize generation as one
Set max_generation:=N
while (generation <= max_generation)
Decode and evaluate chromosomes in the population
Apply selection procedure
Apply genetic operators
```

```
if new chromosomes are better than worse chromosome
in the population
then
    place new chromosome in place of worse chromosome
end if
generation :=generation+1
end while
End
```

3. PARALLEL GENETIC ALGORITHM

A parallel genetic algorithm (PGA) is presented as a solution to the problem of real time versus genetic search encountered in genetic algorithms with large populations [8]. PGA Performing fitness evaluations in parallel will obviously result in an increase in speed of the algorithm roughly proportional to the number of processors used. There are, however, reasons for performing GAs in parallel that are believed to give improved performance [1]. If GA is considered as simply a model of natural systems then some parallel implementations can be viewed as consisting of separate sub-populations evolving independently of each other, with occasional migration allowed between these sub-populations [1]. PGAs can be divided into three general classes, Coarse-grained, Fine-Grained, and Master-Slave.

3.1 The master-slave model

In the master-slave model the master runs the evolutionary algorithm, controls the slaves and distributes the work. The slaves take batches of individuals from the master and evaluate them. Finally send the calculated fitness value back to master.

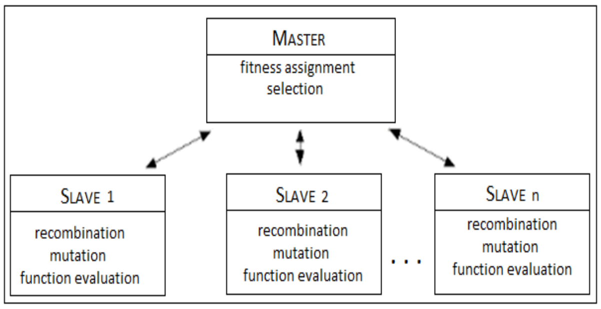

Fig.1 Master-Slave model

3.2 Coarse-grained PGA

The population is divided into a few large subpopulations. Each of these subpopulations are maintained by different processors and some selected individuals are exchangeable via a migration operator. The model is known as Island model or distributed PGA and subpopulation called deme [10,14]. This models are usually implemented on distributed memory MIMD computers[15]. Island model is a popular and effective parallel genetic algorithm [12] and also reduces probability of premature convergence [11] – finding the local instead of the global optimum.

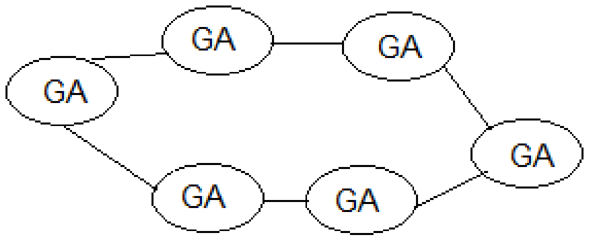

Fig.2 Coarse-grained PGA

3.3 Fine-grained PGA

The population is separated into a large number of very small subpopulations, which are maintained by different processors [8]. The subpopulation may be only an individual. This model is suitable for massively parallel architectures – machines consisting of a huge number of basic processors and connected with a specific high speed topology [8,14]. The computer structure limits an interaction between individuals. This model is machine dependent like Master-Slave PGA.

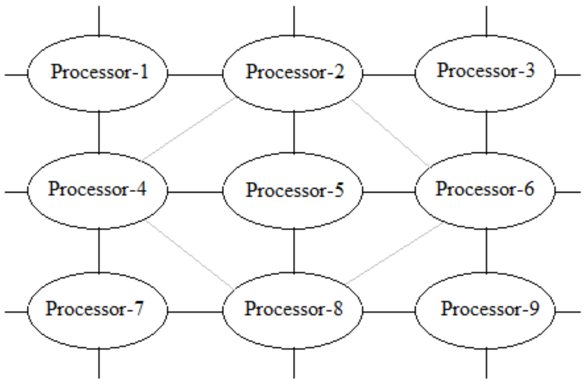

Fig.3 Fine-grained PGA

4. COMPUTATIONAL RESULTS

The proposed parallel genetic uses master-slave GAs parallelization. A master is the main processor store the full population of chromosomes and assigns a certain fraction of the individuals to slave processors, where the slaves evaluate fitness value for the assigned fraction and return their values. The parallelized done only on the fitness evaluation as the fitness of an individual is independent from the rest of the population, and there is no need to communicate during this phase. The crossover probability0.6, mutation probability: 0.01 and tournament size:3. This algorithm has been implemented using java threads in a shared memory environment. For analyzing the performance of the algorithm, different strategies have been used. The result represents average runs among 10 runs and the result is in sec.

Table 1. Comparison of GA and PGA

Population size	**No. of genera- - tions 4, no. of processors 2, no. of tasks:10**		**No. of genera- - tions 8, no. of processors 18, no. of tasks: 24**	
	GA	PGA	GA	PGA
10,000	304	302	287	281
50,000	6520	6404	5895	5765
60,000	8848	8737	7345	7325

5. CONCLUSION

According to the obtained results, the proposed parallel algorithm outperforms the sequential algorithm in case of complex and high number of generation problems. In smaller problems, it is not preferred to use the parallel algorithms. Using the asynchronies may give a better performance since it will never have to wait for all processors to finish their tasks. As a future work, another proposed heuristic, meta-heuristic or evolutionary algorithm could be used. The parallel implementation of the algorithm could be compared with the proposed one, and the results will be compared in terms of accuracy and performance.

6. REFERENCES

[1] *Abtin Hassani* ... *Box 883*. *Västerås*, *Sweden han03007@student.mdh.se*. *Jonatan Treijs*.*Mälardalen's University*. *Box 883*. *Västerås*, *Sweden jts05002@student.mdh.se* ... in *parallel*.

[2] Dhar, V., & Stein, R., Seven Methods for Transforming Corporate Data into Business Intelligence., Prentice Hall 1997, pp. 126-148, 203-210.

[3] Goldberg, D. E., *Genetic and Evolutionary Algorithms Come of Age*, Communications of the ACM, Vol.37, No.3, March 1994, pp.113-119.

[4] Holland, J. H., *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press, 1975.

[5] Kingdon, J., *Intelligent Systems and Financial Forecasting*, Springer Verlag, London 1997.

[6] Medsker,L., *Hybrid Intelligent Systems*, Kluwer Academic Press, Boston 1995.

[7] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin 1996.

[8] Pettey, C.B.Leuze, M.R.Grefenstette, J.J. Parallel genetic algorithm [1987] Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms : July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA.

[9] Coarse-Grained Parallel Genetic Algorithm for Solving the Timetable Problem Shisanu.

[10] D. Andre, J. R. Koza, "Parallel genetic programming on a network of transputers",In Rosca, Justinian (editor), Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications, University of Rochester, National Resource Laboratory for the Study of Brain and Behavior, Technical Report 95-2, June 1995, pp. 111 - 120.

[11] S-C Lin, W.F. Punch and E.D. Goodman, "Coarse- grain Genetic Algorithms, Categorization and New Approaches", Sixth IEEE Parallel and Distributed Processing Oct 1994, pp.28-37.

[12] D.Whitley, S. Rana and R. B. Heckendorn, "Island Model Genetic Algorithms and Linearly Separable Problems",Proceedings of the AISB Workshop on Evolutionary Computation, 1997.

[13] Garey, M. R., Johnson, D. S. 1979. Computers and Intractability, A Guide to The Theory of NP-Completeness, W. H. Freeman and Company.

[14] Abtin Hassani, Jonatan Treijs, "An Overview of Standard and Parallel Genetic Algorithms".

[15] B. S. P. Mishra, S. Dehuri, R. Mall, A. Ghosh, "Parallel Single and Multiple Objectives Genetic Algorithms: A Survey". International Journal of Applied Evolutionary Computation, 2(2), 21-58, April-June 2011 21.