Volume 4, Issue 4, April 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Types and Applications of Parallel Genetic Algorithm

Pradnya S.Borkar
Asst. Prof. Department of Computer Science Engg
Priyadarshini J.L. College of Engg., Nagpur
India

Dr.A.R.Mahajan
HOD, Department of Computer Science Engg.
Priyadarshini Institute of Engg. & Technology, Nagpur
India

Abstract*— Genetic Algorithm (GA) are powerful search techniques that are used to solve various types of problems in various discipline. There are a growing number of programming systems for implementation of genetic algorithms. The potentiality of parallel genetic algorithm is enormous over existing information processing, classification and control of very large and varied data. Parallel genetic algorithm is particularly easy to implement and promise substantial gains in performance. This paper gives a brief overview of advances, computing trends and applications and future perspectives in parallel genetic algorithm.

***Keywords*— Genetic Algorithm, Parallel Genetic Algorithm**

I. INTRODUCTION

Genetic Algorithms (Gas) are search algorithms inspired by genetics and natural selection [1]. Genetic Algorithms in general and Parallel Genetic Algorithms in particular , are of major significance to the development of the new generation of IT applications, The most important phases in Gas are reproduction, mutation, fitness evaluation and selection(Competition). Reproduction is the process by which the genetic material in two or more parent individuals is combine to obtain one or more offspring. Mutation is normally applied to one individual in order to produce a new version of it where some of the original genetic material has been randomly changed. Fitness evaluation is the step in which the quality of an individual is assessed. This is very often the most CPU intensive part of a GA. Selection is an operation used to decide which individuals to use for reproduction and mutations in order to produce new search points. Each individual in the population of a GA is a candidate solution for the problem. A generation of a GA is composed of the following steps

- Fitness-computation
- Selection
- Crossover
- Mutation

Fitness-computation is the competition of individuals and can tell which individual is good for the problem. The selection chooses good individuals to survive and eliminates bad ones. The crossover mates two individuals to produce the next generation individuals. The mutation occurs after crossover so that next generation can be more diverse. With enough generations, GAs can evolve an individual that is the optimal solution to the problem.

GA works as follows:

1. Start with a randomly generated population of n l−bit chromosomes (candidate solutions to a Problem).
2. Calculate the fitness $f(x)$ of each chromosome x in the population.
3. Repeat the following steps until n offspring have been created:

   a) Select a pair of parent chromosomes from the current population, the probability of selection being an increasing function of fitness. Selection is done "with replacement," meaning that the same chromosome can be selected more than once to become a parent.

   b) With probability pc (the "crossover probability" or "crossover rate"), cross over the pair at a randomly chosen point (chosen with uniform probability) to form two offspring. If no crossover takes place, form two offspring that are exact copies of their respective parents.
   (Note that here the crossover rate is defined to be the probability that two parents will cross over in a single point. There are also "multi−point crossover" versions of the GA in which the Crossover rate for a pair of parents is the number of points at which a crossover takes place.)

   c) Mutate the two offspring at each locus with probability pm (the mutation probability or mutation rate), and place the resulting chromosomes in the new population.
   If n is odd, one new population member can be discarded at random.
4. Replace the current population with the new population.

5. Go to step 2.

II. TYPES OF GENETIC ALGORITHM

Basically there are two types of Genetic Algorithm Sequential Genetic Algorithm (Simple GA) and Parallel Genetic Algorithm. Sequential GAs have been shown to be very successful in many applications and in very different domains. However there exist some problems in their utilisation which can all be addressed with some form of Parallel GA(PGA):

- For some kind of problems , the population needs to be very large and the memory required to store each individual may be considerable . In some cases this makes it impossible to run an application efficiently using a single machine, so some parallel form of GA is necessary.
- Fitness evaluation is usually very time-consuming.
- Sequential GAs may get trapped in a sub-optimal region of the search space this becoming unable to find better quality solutions. PGAs can search in parallel different subspaces of the search space, thus making it less likely to become trapped by low-quality subspaces.

Parallel genetic algorithms (PGAs) are parallel stochastic algorithms. Like sequential genetic algorithms (GAs), they are based on the natural evolutionary principle. Better individuals survive and reproduce themselves more often than the worse ones. To speed up the processing of generations of populations, the population can be split into several subpopulations and run them in the parallel way. PGAs have often been applied to various hard optimization problems, machine learning, and prediction problems [3].

III. CLASSIFICATION OF PARALLEL GENETIC ALGORITHM

The basic idea behind most parallel programs is to divide a task into chunks and to solve the chunks simultaneously using multiple processors. Some parallelization methods use a single population, while other divides the population into several relatively isolated subpopulations.

There are three main types of parallel GAs

i) Global single-population Master slave GAs(Micro-grained model)
ii) Single-population fine grained (Cellular model)
iii) Multiple-Population coarse grained GAs (Island Model)

A. Master- Slave Scheme

In the master –slave scheme, as the name specifies one node becomes master and the other nodes become the slaves. The master node maintains the population of GA. The master node assigns individuals to slave nodes to parallelize the calculation of the fitness. The slave nodes calculate the fitness of each individual and sent it back to the master node and then the master nodes does the selection, crossover and mutation.

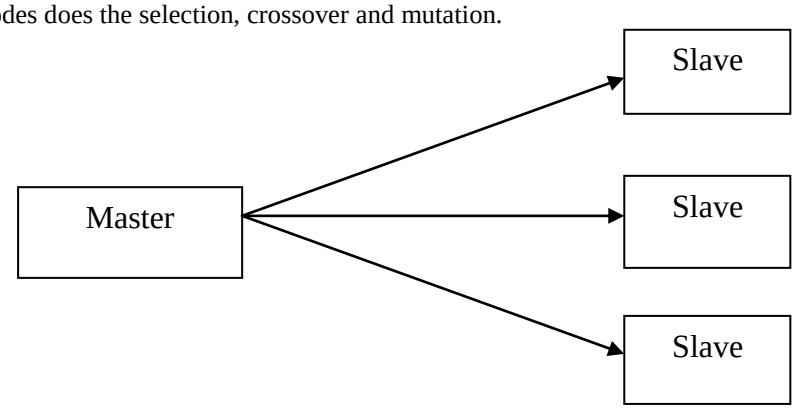

Fig.1 : Master Slave model

B. Cellular Scheme

In this scheme each individual is executed by a computing device separately. Individuals only mate with their connected neighbours during crossover and mutation. The neighbours of individuals are decided by the connection topology. The Cellular scheme is a fine grained parallel scheme.

C. Island Scheme

In the Island scheme, the population of GA is divided into several sub-population called islands. The sub-population works in isolation from each other. Island communicate and exchange some of their best individuals, this process is called migration. The migration helps to spread the local best solutions in islands to other islands, with this the population can evolve to better solutions. The island scheme is perfectly suitable for the parallel and distributed environments. With this scheme, the computation load can be distributed to different computing devices without huge overheads caused by the communication between islands. Islands are connected by migration in a ring topology. Each island maintains a migration data structure to support the migration. When an island does a migration it puts a fixed number of individuals into its neighbour's migration data structure. Each Island gets individuals from its migration data structure and then the migration step is complete. There are two types of islands scheme Synchronous Island and Asynchronous.

In synchronous island scheme, after achieving a fixed number of generations, a migration takes place. It means that a faster island has to wait for the slower island to reach up to the fixed number of generations. Whereas in Asynchronous scheme, Islands do not have to wait for each other, though they do the migration after every fixed number of generations [4].

IV. RELATED WORK USING PARALLEL GENETIC ALGORITHM

In[8] P. vidal and E. Alba presented a novel implementation of a Cellular Genetic algorithm model for a multi-GPU platform using NVIDA'S CUDA technology . this multi- GPU model is compared first against a serial version in CPU and then versus an implementation on a single GPU. They divide the different operations of the GA into distinct sets of instructions called kernels. Using the multi-GPU platform ,they observ that the speedup with respect to the CPU version ranges from 8 to 771,while it is similar to that of the GPU,with a little overhead in the multi-GPU case. Their results demonstrate that multi-GPU desktops can serve as cost-effective parallel computing platforms to obtain accurate results in very short time.

Two kinds of parallel genetic algorithm(PGA) are implemented in [9] based on MATLAB parallel computing Toolbox ™ and distributed computing Server ™ Software. Parallel for-loops, SPMD(Single Program Multiple Data) block and co-distributed arrays, three basic parallel programming modes in MATLAB are employed to accomplish the global and coarse-grained PGAs. To validate and compare the implementation, both PGAs are applied to run the problem of range image registration. The differential evolution and genetic algorithms implemented on CUDA in [10] and compare when solving the independent tasks scheduling problem. In such environment, the nature inspired metaheuristics can be in suitable cases implemented in parallel without additional costs. Genetic algorithm involves a number of other operations which, if parallelized properly, may also end up with a better parallelization to an existing serial GA implementation in [11] Parallelization of binary and real-coded genetic algorithms for the CUDA platform using its C API extension. The bottlenecks in the GA algorithms for a parallel implementation are identified and modified suitably. The results are compared with the serial algorithm ion accuracy and clock time for varying problems by studying the effect of a number of parameters such as i) differing population sizes (ii) differing number of threads(iii) differing problem sizes and (iv) problems having differing complexities, such as evaluation time and interactions among variables. In[12], Authors model the resources allocation for multicores as an optimization space, including variant selection, grouping and PE assignment. Finding efficient parallelization schemes for a multi-task program is to explore this search space and hence the genetic algorithm is used to find out the performance optimal solutions. In[13] introduces a novel implementation of the genetic algorithm exploiting a multi-GPU cluster. The proposed implementation employs an island-based genetic algorithm where every GPU evolves a single island. The individuals are processed by CUDA warps, which enable the solution of large knapsack instances and eliminates undesirable thread divergence. The MPI interface is used to exchange genetic material among isolated islands and collect statistical data. The characteristics of the proposed GAs are investigated on a two-node cluster composed of 14 Fermi GPUs and 4 six-core Intel Xeon processors. The overall GPU performance of the proposed GA reaches 5.67 TFLOPS.

The [14] deals with the mapping of the parallel island-based genetic algorithm with unidirectional ring migrations to nVidia CUDA software model. The proposed mapping is tested using Rosen-brock's, Griewank's and Michalewicz's benchmark functions. The ob-tained results indicate that our approach leads to speedups up to seven thousand times higher compared to one CPU thread while maintaining a reasonable results quality. This clearly shows that GPUs have a potential for acceleration of GAs and allow solving much complex tasks.

In[15] , it is difficult to find the solutions which satisfy all objective functions because of their trade-off. Especially when there are many objective functions, it is obvious that it needs a lot of time to search for effective Pareto solutions and find them. This paper proposes the interactive way of addition and deletion of islands to the original ones based on user's requirements with the visualization of acquired solutions in island model for MOGA. This paper applies the proposed method to Nurse Scheduling Problem (NSP) using the visualization by Principal Component Analysis (PCA). Through the experiment, it is confirmed that an interactive tuning of the weights for the objective functions leaded to the acquisition of better Pareto solutions which a user wants while they are difficult to be acquired by the prepared weights.

V. TRENDS IN COMPUTING

The trends and fundamental issues in parallel genetic computing is related to the designs, implementation and characteristics of parallel algorithms. Architecture is one of the important issue because the working of PGA is mainly depends on the architecture. The various architectural models are SISD, SIMD, MISD, and MIMD. For working purpose of PGA mainly SIMD and MIMD is important.

VI. APPLICATIONS

Applications of PGAs(GAs) are regularly wide and range from Numerical Mathematics and Graph Theory(numerical function optimization, graph bipartity, graph partitioning problem, scheduling problems, mission routing problems),through computer science(searching for weights of neural networks, optimization of server load or database queries), Finance and Economies(financial balancing problems, transport problems, modeling systems, prediction of time series)to Technology and Engineering(Optimization of VLSI circuits, optimization of car wheels, optimization in Material Engineering[3].

VII. CONCLUSION

PGAs(GAs) are required to meet the need of various applications like data mining, aeronautical Engineering, Network theory, Biomedical Science as large data set is to be processed. This article gives the overview of PGAs (GAs) types, Computing Trends and Applications.

REFERENCES

[1] David Goldberg, Genetic Algoritms in search, Optimization and Machine Learning, Addison-wesley, Reading MA, 1989.

[2] Mariusz Nowostawski, Riccardo Poli "Parallel Genetic Algorithm Taxonomy", KES'99,May 13,1999.

[3] Zdenek Konfrst "Parallel Genetic Algorithms: advacnes, Computing Trends, Applicatons and Perspectives" Proceedings of the 18th International Parallel and Distributed Processing Symposium(IPDPS'04),2004 IEEE.

[4] Erick Cantu-Paz" A Survey of Parallel Genetic Algorithm".

[5] P.Vidal and E. Alba "*A multi-gpu implementation of a cellular genetic algorithm*" in 2010 IEEE Congress on Evolutionary Computation (CEC),July 2010,pp. 1-7.

[6] Li Nan, Gao Pengdong,Lu Yongquan, Yu Wenbua "*The implementation and comparison of Two kinds of Parallel Genetic Algorithm Using Matlab*." 2010,IEEE Ninth International Symposium on Distributed Computing and Applications to Business,Engineering and Science.,page(s) 13-17,2010.

[7] Pavel Kromer,Jan Platos,Vaclav Snasel,Ajith Abraham,"*A Comparison of Many-threaded Differential Evolution and Genetic Algorithms on CUDA"*, Third world congress on Nature and Biologically Inspired Computing(NaBIC) ,IEEE,2011 page(s) : 509-514.

[8] Ramnik Arora, Rupesh Tulshyan, Kalyanmoy Deb"*Parallelization of Binary and Real-coded Genetic Algorithms on CUDA*",IEEECongress on Evolutionary Computation (CEC),Page(s):1-8.

[9] Miao Wang, Nicolas Benoit,Francois Bodin, Zhiying Wang "*Model Driven Iterative Multi-Dimensional Parallelization of Multi-task programs for the Cell BE*: A Genetic Algorithm-based Approach", 2010,IEEE 18th Euromicro Conference on Parallel,Distributed and Network-based Processing.

[10] Jiri Jaros,"*Multi Gup Based Implementation of Island Model*",WCCI, 2012 IEEE World Congress on Computational Intelligence June, 10-15, 2012, Brisbane, Australia.

[11] Petr Pospichal, Jiri Jaros and Josef Schwar"*Parallel Genetic algorithm on the CUDA Architecture*", EvoApplications 2010,Part 1,LNCS 6024,pp 442-451,Springer-verlag Berlin Heiderlberg, 2010.

[12] Tomohiro Y. Takeshi F."*Study on Effect of MOGA with Interactive Island Model Using Visualization*",2010 IEEE Congress on Evolutionary computation(CEC),18-23 July 2010,Page(s) 1-6.

