A NEW MODEL OF GLOBAL PARALLEL GENETIC ALGORITHM

Marin Golub, Domagoj Jakobović
Faculty of Electrical Engineering and Computing, University of Zagreb
Department of Electronics, Microelectronics, Computer and Intelligent Systems
Unska 3, 10000 Zagreb, Croatia
e-mail: marin.golub@fer.hr, domagoj.jakobovic@fer.hr

Abstract: *In this paper we describe a multithreaded parallel genetic algorithm (PGA) implementation. Considering the basic models of parallel genetic algorithms, we identify a variant of global PGA (GPGA) as the most appropriate one for use on a multiprocessor system with few processors. The difference between the synchronous and asynchronous model is analized and their characteristics are evaluated. Unlike some authors, we choose not to allow a single individual to be engaged in a tournament competition in more than one instance (no duplicates). The probability of selection for elimination of an individual is than determined based on the fitness of the chromosome and compared with the same probability of the duplicate-allowing algorithm. Finally, main advantages and disadvantages of our GPGA as well as performance comparison with sequential GA are stated.*

Key words: parallel genetic algorithm, multithreading, tournament selection

1. Introduction

As genetic algorithms usually require more computation time than other heuristic approaches, the basic motivation of GA parallelization is the reduction of the processing time needed to reach an acceptable solution. A good method of parallelization should preserve any properties that sequential algorithm with the same genetic operators would have. It should also not introduce too many additional parameters whose values could significantly affect the GA performance. Furthermore, it is highly desirable to eliminate any need for process intercommunication and synchronization. That is why the asynchronous approach has been favoured in the majority of applications.

In this work, the multithreading technique is recognised as an efficient tool for transforming the genetic algorithm into parallel form. Multiple threads are relatively simple to implement and they require less preparation and handling than processes.

2. A Short Survey of Parallel Genetic Algorithms

Existing parallel implementations of genetic algorithm can be classified into three main types of PGAs [Cantú-Paz, 1998a; Logar *at al.*, 1997; Michalewicz, 1994] (*Figure 1*): global single-population master-slave genetic algorithms (GPGA), massively parallel genetic algorithms (MPGA) and distributed genetic algorithms (DGA). Furthermore, there are trivial parallel genetic algorithm (TPGA) and two additional models of PGAs which combine three main types of PGAs or combine PGA with some other optimization method: hierarchical parallel genetic algorithms (HPGA) and hybrid parallel genetic algorithms.

Trivial parallel genetic algorithm or embarrassingly parallel genetic algorithm uses available processors to execute independent problems. There is no communication between the different processes. This extremely simple method is useful for gathering statistics. For example, it can be used for finding optimal set of parameters: several copies of the same

problem run on the same number of processors, but with different GA parameters [Tomassini, 1999].

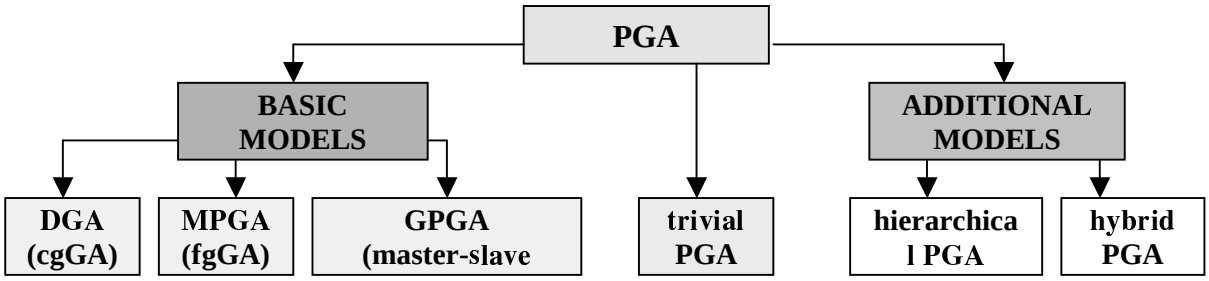

Figure 1. ***Models of parallel genetic algorithm***

Global parallel genetic algorithms or master-slave genetic algorithms consist of one population, but evaluation of fitness is distributed among several processors (*Figure 2*).

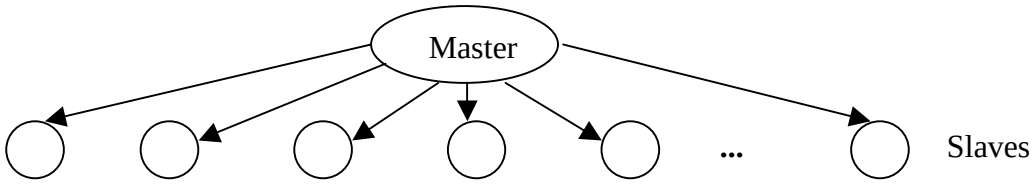

Figure 2. ***Master-slave genetic algorithm***

As in the serial genetic algorithm, selection and mating are global: each individual may compete and mate with any other. The evaluation of the individuals is usually parallelized, because the fitness of an individual is independent from the rest of the population and there is no need to communicate during this phase. Communication occurs only as each slave receives its subset of individuals to evaluate and when the slaves return the fitness values.

The GPGA is synchronous if the algorithm stops and waits the fitness values for all the population before proceeding into the next generation. Synchronous GPGA has the same properties as sequential genetic algorithm, but it is faster if the algorithm spends most of the time for the evaluation process. Synchronous master-slave GAs have many advantages: they explore the search space exactly as a sequential GA, they are easy to implement and significant performance improvements are possible in many cases [Cantú-Paz, 1998b].

Massively parallel genetic algorithms are also called fine-grained genetic algorithms (fgGA) and they are suited for massively parallel computers (*Figure 3*) such as MasPar MP-1 [Logar *at al.*, 1992; Sarma *at al.*, 1996].

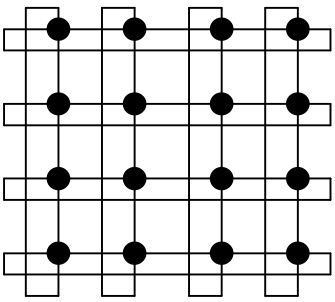

Figure 3. ***A torus of 16 processors***

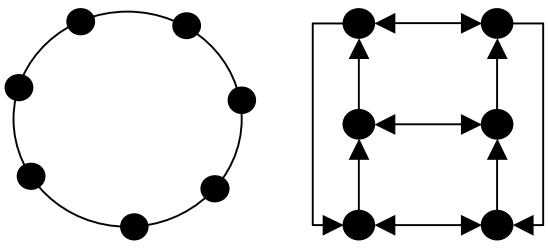

Figure 4. ***A schematic of DGA with ring (left) and ladder topology (right)***

MPGA has one population and interactions between individuals is limited: selection and mating are restricted to a small neighborhood. The overlapping neighborhoods allow individuals to move around continuously and it provides an implicit mechanism for migration. A good solution can disseminate across the entire population. New parameters are neighborhood size and shape.

Distributed genetic algorithms are the most popular parallel methods. Such algorithms assume that several subpopulations (demes) evolve in parallel and that is why this PGA is also called multiple-population or multiple-demes genetic algorithm.

The models include a concept of migration (movement of an individual string from one subpopulation to another). It uses multiple demes (populations) that occasionally exchange some individuals in a process called migration. A specification of an island GAs defines the size and number of demes, the topology of the connections between them (*Figure 4*), the migration rate (the fraction of the population that migrates), the frequency of migrations and the policy to select emigrants and to replace existing individuals with incoming migrants. All these seven new parameters have a great influence on the quality of the search and on the efficiency of the algorithm [Cantú-Paz, 1999]. Because they are controlled by many parameters, the multiple-population PGAs are the hardest to use.

Hierarchical parallel genetic algorithms combine two of three basic models of PGA. When two methods of parallelizing genetic algorithms are combined they form a hierarchy. This class of algorithms are called hierarchical because at higher level they are multiple-deme algorithms with master-slave or fine grained at the lower level. Also, at both levels can be multiple-deme genetic algorithms. Hierarchical implementations can reduce the execution time more than any of their components alone [Cantú-Paz, 1998a]. *Hybrid parallel genetic algorithms* combine PGA with some classical optimization method, for example local hill-climbing [Muehlenbein, 1991].

3. A New Model of GPGA

Which one of three basic models is the most suitable for implementation on a shared memory multiprocessor system with few processors? Obviously the massively PGA is not suitable, because it needs massively parallel computers with a number of processors (several hundreds or even thousands). Two demes (if we have two processor system) is too small number of subpopulations for the distributed genetic algorithm. Even if we have more than two processors, there is still too many new parameters that we must set: migration rate, frequency of migrations, the policy of migrants selection and the topology.

In contrast of other basic models of PGA the operation of GPGA is identical to a serial GA, and therefore any available knowledge about serial GAs can be applied directly to global PGA [Cantú-Paz *at al.*, 1999]. In the traditional master-slave model, the master processor stores the entire population and applies genetic operators to produce the next generation. The slave processors are used to evaluate the fitness of a fraction of the population in parallel.

In our implementation of master-slave GA, the master creates random initial population, evaluates created individuals and starts the slaves (*Figure 5*). Each slave performs whole evolution process in contrast of traditional master-slave GA where the slaves only evaluate the fitness. This is a model of global GA, because each individual may compete and mate with any other.

In our case, the algorithm has elimination selection and two or more threads in each iteration should not eliminate the same individual. So, the elimination of individuals can be

synchronized. This model of GPGA, if it is synchronous, has exactly the same properties as the same sequential GA, with speed being the only difference. The asynchronous GPGA with elimination selection (*Figure 5*), on the other hand, does not have the same properties as sequential GA, but the expected speed-up factor is equal to the number of processors. The worst thing that could happen when we implement asynchronous GPGA is that two or more threads select the same individual for elimination. The work of only one of them will take effect, while other threads will work in vain. The total number of iterations is smaller than given number of iterations, because some threads spend some iterations in vain. That is the reason why the algorithm does not give so good solution as the same sequential GA in given number of iterations. But if we know the probability of multiple elimination of the same individual by several threads at the same time, we can calculate the expected number of iterations when threads will perform genetic operators in vain. If the total number of iterations is increased, the asynchronous GPGA will act exactly as, but faster than, sequential GA.

```
Master thread{
  initialize population;
  evaluate population;
  for(i=1;i<NUMBER_OF_PROCESSORS;i++){
     create new Slave thread;
  }
}
```

```
Slave thread{
  while(termination criterion is not reached){
    select three different individuals;
    eliminate the worst individ. of 3 selected;
    child=crossover(survived two individuals);
    replace deleted individual with child;
    perform mutation with probability pm;
    evaluate new individual;
  }
}
```

Figure 5. ***Asynchronous global parallel genetic algorithm with 3-tournament bad individual selection***

4. The Probability of Selection

The tournament selection is suitable for parallel execution [Cantú-Paz *at al.*, 1999, Cantú-Paz 1999, Yoshida *at al.*, 1999]. The 3-tournament bad individual selection in each step of the evolution chooses with equal probability three individuals from the mating pool. Then, it eliminates the weakest one of those three individuals. The survived two individuals are parents of a child which will replace the eliminated one (*Figure 5*).

Let the individuals be indexed by their fitness value, i.e. the best one has the index i=1 and the worst one i=N, where N is population size. The probability $p_k(i)$ of selection for elimination of the i-th individual, where i=3,4,...,N, with k-tournament bad individual selection is given by:

$$p_k(i) = \frac{\binom{i}{k} - \binom{i-1}{k}}{\binom{N}{k}}. \qquad (1)$$

Selection of the same individual more than once in one tournament cycle is not possible. On the other hand, Bäck, Miller, Goldberg, Blickle and Thiele use tournament selection which in the same iteration can select the same individual more then once [Bäck 1994, Miller *at al.*, 1995, Blickle *at al.*, 1995]. In this case, the probability $p_k'(i)$ of selection for elimination of the i-th individual is given by:

$$p_k'(i) = \left(\frac{i}{N}\right)^k - \left(\frac{i-1}{N}\right)^k. \qquad (2)$$

At the first sight, it would seem that tournament selection without duplicates is more complex and time consumant. Besides, the characteristic values such as selection intensity, reproduction rate and loss of diversity have already been determined for duplicate tournament selection. Genetic algorithm's behaviour can be fairly predicted using those values. However, if no control mechanism is applied, elimination tournament selection can produce additional duplicates, which only slow down the evolution process. Such a mechanism, an example of which can be found in [Michalewicz, 1994], eliminates duplicate individuals, but spends much more computation time then relatively simple duplicate check in a single tournament. Furthermore, the elitism is not inherently implemented, unlike in no-duplicates tournament selection, which can be percieved from (2) if i equals 1. Although the expressions for selection probability differ significantly, the actual distributions are very similar, as shown in Figure 6.

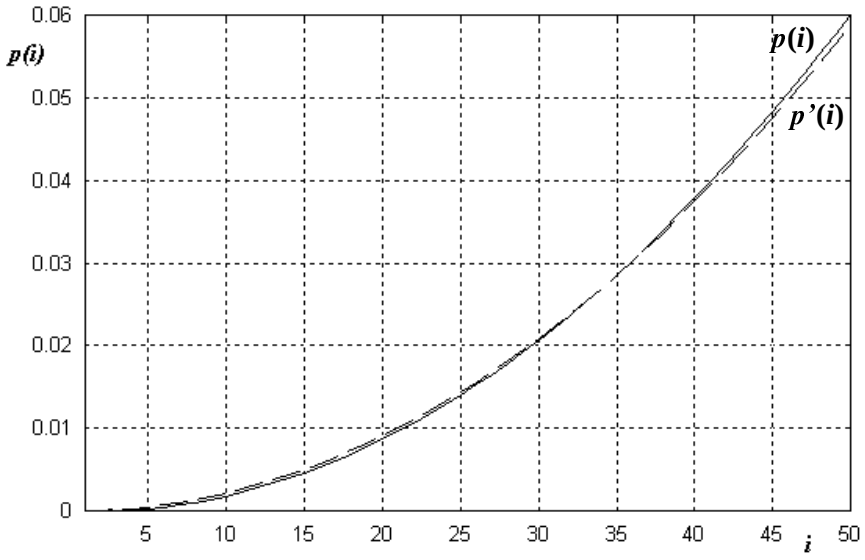

Figure 6. ***Distribution of selection for elimination probability with 3-tournament selection and N=50***

5. Concluding remarks

The presented model of global parallel genetic algorithm is suitable for implementation on a shared memory computer with several processors. The difference between traditional GPGA (master-slave GA) and the described GPGA is in tasks which master and slaves perform. In the traditional GPGA slaves only evaluate individuals, while the master distributes individuals among slaves for evaluation and the master performs all genetic operators. The assumption is that the evaluation takes most of the time in the evolution process. In our case, the master only initializes the population, while slaves perform the whole evolution process including evaluation. The parts obtained by dividing the genetic algorithm are independent. The critical sections are avoided, because the synchronization mechanisms would slow down the parallel program.

Advantages of our approach are: the algorithm is simple for implementation, the elitism is inherently implemented, all genetic operators and evaluation is parallelized, there is no need for any communication mechanism (the whole population is placed into shared memory), it works without any synchronization, the execution time for a given number of iterations does not depend on the population size N for a constant number of iterations and speed-up factor is

near the number of processors. Moreover, the algorithm can be executed by any given number of processors without any code adaptation. Disadvantages are: the number of iterations must be increased because some threads may perform invalid iterations and it is not suitable for any fitness-proportional selection [Cantú-Paz *at al.*, 1999].

Acknowledgement

This work was carried out within the project 036-014 *Problem-Solving Environments in Engineering*, funded by Ministry of Science and Technology of the Republic of Croatia.

References:

1. Bäck, T., "Selective Pressure in Evolutionary Algorithms: A Characterization of Selection Mechanisms", *Proc. of the First IEEE Conference on Evolutionary Computation*, IEEE Press, Piscataway NJ, pp. 57-62, 1994., available from: http://ls11-www.informatik.uni-dortmund.de/people/baeck/papers/wcci94-sel.ps.gz.
2. Blickle, T., Thiele, L., "A Mathematical Analisys of Tournament Selection", *Proc. of the Sixth International Conference on Genetic Algorithms*, San Francisco, CA, pp. 2-8, 1995.
3. Cantú-Paz, E., "A Survey of Parallel Genetic Algorithms", *Calculateurs Paralleles*, Vol. 10, No. 2. Paris: Hermes, 1998., available via ftp from: ftp://ftp-illigal.ge.uiuc.edu/pub/papers/Publications/cantupaz/survey.ps.Z.
4. Cantú-Paz, E., "Designing Efficient Master-slave Parallel Genetic Algorithms", *Genetic Programming: Proc. of the Third Annual Conference*, San Francisco, CA, 1998., pp. 455-460.
5. Cantú-Paz, E., Goldberg, D.E., "Parallel Genetic Algorithms with Distributed Panmictic Populations", 1999., available from: http://www-illigal.ge.uiuc.edu/cgi-bin/orderform/orderform.cgi.
6. Cantú-Paz, E., "Migration Policies, Selection 0Pressure, and Parallel Evolutionary Algorithms", 1999., available via ftp from: ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs.
7. Logar, A.M., Corwin, E.M., English, T.M., "Implementation of massively parallel genetic algorithms on the MasPar MP-1", *Applied computing*, technological challenges of the 1990's, Vol. II, page 1015, 1992.
8. Michalewicz, Z. (1992), *Genetic Algorithms + Data Structures = Evolutionary Programs*, Springer-Verlag, Berlin.
9. Miller, B.L., Goldberg, D.E., "GAs Tournament Selection and the Effects of Noise", 1995., available from: http://www.dai.ed.ac.uk/groups/evalg/Local_Copies_of_Papers/Miller.Goldberg.GAs_Tournament_Selection_and_the_Effects_of_Noise.ps.gz.
10. Muehlenbein, H., "Evolution in Time and Space - The Parallel Genetic Algorithm", *Foundations of Genetic Algorithms*, G. Rawlins (ed.), pp. 316-337, Morgan-Kaufman, 1991., available via ftp from: ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-91_01.ps.
11. Sarma, J., De Jong, K., "An Analysis of the Effects of Neighborhood Size and Shape on Local Selection Algorithms", *Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature (PPSN96)*, Sept. 22-26, Berlin, Germany, 1996.
12. Tomassini, M., "Parallel and Distributed Evolutionary Algorithms: A Review", in *Evolutionary Algorithms in Engineering and Computer Science*, editors: Miettinen, K.,

Neittaanmaki, P., Makela, M.M., Periaux, J., John Wiley & sons, LTD, 1999., pp. 113-131.

13. Yoshida, N., Yasuoka, T., Moriki, T., "Parallel and Distributed Processing in VLSI Implementation of Genetic Algorithms", *Proceedings of the Third International ICSC Symposia on Intelligent Industrial Autimation*, IIA'99 and Soft Computnig, SOCO'99, June 1-4, 1999., Genova, Italy, pp. 450-454.