A SURVEY ON SYNCHRONOUS AND ASYNCHRONOUS PARALLEL DISTRIBUTED ALGORITHMS

Ajitesh S. Baghel*

Rakesh Kumar Katare*

Abstract. In a parallel hereditary calculation (PGA) a few conveying nodal GAs advance in parallel to tackle the same issue. PGAs have been customarily used to amplify the force of serial GAs since they frequently can be customized to give a bigger effectiveness on complex hunt undertakings. This has prompted a significant number of distinctive models and executions that block direct correlations and learning trade. To fill this crevice we start by giving a typical system to mulling over PGAs. This permits us to examine the significance of the synchronism in the movement venture of parallel circulated GAs. We will demonstrate how this usage issue influences the assessment exertion and in addition the pursuit time and the speedup. Likewise, we consider prevalent development plans of panmictic (relentless state) and organized populace (cell) GAs for the islands. The assessed PGAs show direct and even super-straight speedup when keep running in a group of workstations. They likewise indicate essential numerical advantages when contrasted and their successive partners. Moreover, we generally report lower quest times for the nonconcurrent forms.

Keywords: ***GA,PGA,MIMD***

* Dept. of Computer Science A.P.S.Univesity, Rewa (M.P.)

1. Introduction:

Genetic algorithms (GAs) are stochastic search methods that have been successfully applied in many search, optimization, and machine learning problems [2]. Unlike most other optimization techniques, GAs maintain a population of encoded tentative solutions that are competitively manipulated by applying some variation operators to find a global optimum. A sequential GA proceeds in an iterative manner by generating new populations of strings from the old ones. Every string is the encoded (binary, real, ...) version of a tentative solution. An evaluation function associates a fitness measure to every string indicating its suitability to the problem. The canonical GA applies stochastic operators such as selection, crossover, and mutation on an initially random population in order to compute a whole generation of new strings.

For non-trivial problems this process might require high computational resources, and thus, a variety of algorithmic issues are being studied to design efficient Gas. With this goal in mind, numerous advances are continuously being achieved by designing new operators, hybrid algorithms, and more. We extend one of such improvements consisting in using parallel models of GAs (PGAs).

Several arguments justify our work. First of all, PGAs are naturally prone to parallelism since the operations on the strings can be easily undertaken in parallel.

The evidences of a higher efficiency [3], larger diversity maintenance, additional availability of memory/cpu, and multi-solution capabilities, reinforce the importance of the research advances with PGAs [9].

Using a PGA often leads to superior numerical performance and not only to a faster algorithm [5]. However, the truly interesting observation is that the use of a structured population, either in the form of a set of islands [3] or a diffusion grid [11], is the responsible of such numerical benefits. As a consequence, many authors do not

use a parallel machine at all to run structured-population models, and still get better results than with serial GAs [5].

In traditional PGA works it is assumed that the model maps directly onto the parallel hardware, thus making no distinction between the model and its implementation. However, once a structured-population model has been defined it can be implemented in any monoprocessor or parallel machine. This separate vision of model vs. implementation raises several questions. Firstly, any GA can be run in parallel, although a linear speedup is not always possible. In this

sense, our contribution is to extend the existing work on distributed GAs (we will use dGA for short) from generational island evolution to steady-state [10], and cellular GAs [11].

Secondly, this suggests the necessity of using a difficult and heterogeneous test suite. In this sense, we use a test suite composed of separable, non-separable, multimodal, deceptive, and epistatic problems (main difficulties of real applications). Thirdly, the experiments should be replicable to help future extensions of our work. This led us to using a readily available parallel hardware such as a cluster of workstations and to describe the references, parameters, and techniques.

Finally, some questions are open in relation to the physically parallel execution of the models. In our case we study a decisive implementation issue: the synchronism in the communication step [6]. Parameters such as the communication frequency used in a PGA are also analyzed in this paper.

The high number of non-standard or machine-dependent PGAs has led to efficient algorithms in many domains. However, these unstructured approaches often hide their canonical behavior, thus making it difficult to forecast further behavior and to make knowledge exchange. This is why we adopt a unified study of the studied models.

2. A Common Framework for Parallel Genetic Algorithms:

In this section we briefly want to show how coarse (cgPGA) and fine grain (fgPGA) PGAs are subclasses of the same kind of parallel GA consisting in a set of communicating sub-algorithms. We propose a change in the nomenclature to call them distributed and cellular GAs (dGA and cGA), since the grain is usually intended to refer to their computation/communication ratio, while the actual differences can be also found in the way in which they both structure their populations (see Figure 1[2]).While a distributed GA has large sub-populations (>>1) a cGA has typically one single string in every sub-algorithm. For a dGA the sub-algorithms are loosely connected, while for a cGA they are tightly connected. In addition, in a dGA there exist only a few sub-algorithms, while in a cGA there is a large number of them.

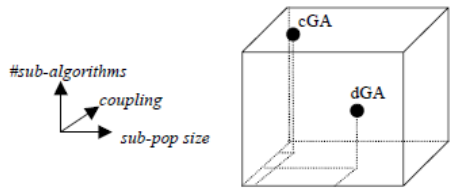

Figure 1. The Structured-Population Genetic Algorithm Cube

This approach to PGAs is *directly* supported in only a few real implementations of PGA software. However, it is very important since we can study the full spectrum of parallel implementations by only considering different coding, operators, and communication details. In fact, this is a natural vision of PGAs that is inspired in the initial work of Holland, in which a *granulated adaptive system* is a set of grains with shared structures. Every grain has its own reproductive plan and internal/external representations for its structures.

This communication usually consists in exchanging a set of individuals or population statistics. All the sub-algorithms are though to perform the same reproductive plan. Otherwise the PGA is heterogeneous [2].

In particular, our steady-state panmictic algorithm (Figure 2a[4]) generates one single individual in every iteration. It is inserted back in the population only if it is better (larger fitness than) the worst existing individual.

ALGORITHM 1: PARALLEL GENETIC ALGORITHM

$t := 0;$

initialize: $P(0) := \{\vec{a}_1(0),...,\vec{a}_\mu(0)\} \in I^\mu;$

evaluate: $P(0) : \{\Phi(\vec{a}_1(0)),...,\Phi(\vec{a}_\mu(0))\};$

while not $t(P(t))$ **do** *// Reproductive Loop*

select: $P'(t) := s_{\theta_s}(P(t));$

recombine: $P''(t) := \otimes_{\theta_c}(P'(t));$

mutate: $P'''(t) := m_{\theta_m}(P''(t));$

evaluate: $P'''(t) : \{\Phi(\vec{a}_1'''(0)),...,\Phi(\vec{a}_\lambda'''(0))\};$

replace: $P(t+1) := r_{\theta_r}(P'''(t) \cup Q);$

<communication step>

$t := t+1;$

end while

In all the cGAs we use (Figure 2b) a NEWS neighborhood is defined (**N**orth-**E**ast- **W**est-**S**outh

in a toroidal grid [11]) in which overlapping demes of 5 strings (4+1) execute the same reproductive plan. In every deme the new string computed after selection, crossover, and mutation replaces the current one only if it is better.

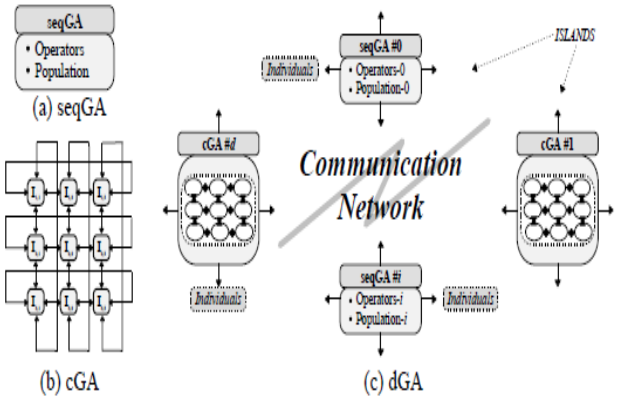

Figure 2. Two basic non-distributed models (a) (b), and a possible -merged!- distribution (c)

In all this paper we deal with MIMD implementations (Figure 2c) of homogeneous dGAs in which migrants are selected *randomly*, and the target island replaces its *worst* string with the incoming one only if it is better. The sub-algorithms are disposed in a unidirectional ring (easy implementation and other advantages).

III. Real Time Analysis of the Impact of the Synchronism:

Synchronous and asynchronous parallel dGAs perform both the same algorithm. However, sync islands wait for every incoming string they must accept, while a sync ones do not. In general, no differences should appear in the search, provided that all the machines are of the same type (our case). On the contrary, the execution time may be modified due to the continuous waits induced in a synchronous model.

Before discussing the results on the speedup of the analyzed algorithms, we need to make some considerations. As many authors have established [6], sequential and parallel GAs *must* be compared by running them until a solution of the same quality has been found, *and not* until the same number of steps has been completed. In deterministic algorithms the speedup (Equation 3) is upper bounded by the number of

processors *nproc*, but for a PGA it might not, since it reduces *both* the number of necessary steps *and* the expected execution time *Tnproc* in relation to the sequential one *T*1. This means that

super-linear speedups are possible [9].

$$s(n_{nproc}) = \frac{T_1}{T_{n_{proc}}} \quad (3)$$

IV. Concluding Remarks:

In this paper we have stated the advantages of using parallel distributed GAs and of performing a common analysis of different models. Parallel distributed GAs almost always outperformed sequential GAs. We have distributed both, a sequential steadystate GA, and a cellular GA with the goal of extending the traditional studies. In all the asynchronous algorithms outperformed their equivalent synchronous counterparts in real time. This confirms other existing results with different PGAs and problems, clearly stating the advantages of the asynchronous communications [6].

V. References:

1. Alba E., Aldana J. F., Troya J. M.: "Genetic Algorithms as Heuristics for Optimizing ANN Design". In Albrecht R. F., Reeves C. R., Steele N. C. (eds.): *Artificial Neural Nets and Genetic Algorithms*. Springer-Verlag (1993) 683-690.

2. Bäck T., Fogel D., Michalewicz Z. (eds.): *Handbook of Evolutionary Computation*. Oxford University Press (1997).

3. Belding T. C.: "The Distributed Genetic Algorithm Revisited". In Eshelman L. J. (ed.): *Proceedings of the Sixth International Conference on Genetic Algorithms*. Morgan Kaufmann, San Francisco, CA (1995) 114-121.

4. Cammarata G., Cavalieri S., Fichera A., Marletta L.: "Noise Prediction in Urban Traffic by a Neural Approach". In Mira J., Cabestany J., Prieto A. (eds.): *Proceedings of the International Workshop on Artificial Neural Networks*, Springer-Verlag (1993) 611-619.

5. Gordon V. S., Whitley D.: "Serial and Parallel Genetic Algorithms as Function Optimizers". In Forrest S. (ed.): *Proceedings of the Fifth International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA (1993) 177-183.

6. Hart W. E., Baden S., Belew R. K., Kohn S.: "Analysis of the Numerical Effects of Parallelism on a Parallel Genetic Algorithm". In IEEE (ed.): CD-ROM IPPS97 (1997).

7. Jelasity M.: "A Wave Analysis of the Subset Sum Problem". In Bäck T. (ed.): *Proceedings of the Seventh International Conference on Genetic Algorithms*. Morgan Kaufmann, San Francisco, CA (1997) 89-96.

8. Romaniuk, S. G.: "Evolutionary Growth Perceptrons". In Forrest S. (ed.): *Proceedings ofthe Fifth International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA (1993) 334-341.

9. Shonkwiler R. "Parallel Genetic Algorithms". In Forrest S. (ed.): *Proceedings of the Fifth International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA (1993) 199-205.

10. Syswerda G.: "A Study of Reproduction in Generational and Steady-State Genetic Algorithms". In Rawlins G. (ed.): *Foundations of GAs*, Morgan Kaufmann (1991) 94-101.

11. Whitley D.: "Cellular Genetic Algorithms". In Forrest S. (ed.): *Proceedings of the Fifth International Conference on GAs*. Morgan Kaufmann, San Mateo, CA (1993) 658.