Parallel Genetic Algorithms –State Of The Art Survey

P. Visalakshi[1], T. Hamsapriya[2], S.N. Sivanandam[3]

ABSTRACT

Genetic algorithms (GAs) are powerful search techniques based on the mechanics of natural selection and natural genetics that are used successfully to solve problems in many different disciplines. In this work we review the most important existing developments in the class of Parallel Genetic Algorithms (PGAs). An exceptional characteristic of PGAs is that, they are not just the parallel version of a sequential genetic algorithm intended to provide speed gains but also represents a new kind of meta-heuristics of higher efficiency and efficacy. The good robustness of these algorithms on problems of high complexity has led to an increasing number of applications in the fields of artificial intelligence, numeric and combinatorial optimization, business, engineering, etc.

Keywords: Parallel genetic algorithms (PGAs), evolution topics, global single-population master slave GAs, single-population fine-grained GAs and multiple-population coarse grained GAs.

1. INTRODUCTION

Sequential GAs are generally able to find good solutions in reasonable amounts of time, but as they are applied to harder and bigger problems there is an increase in the time required to find adequate solutions. As a consequence, there have been multiple efforts to make GAs faster, and one of the most promising choices is to use parallel implementations. Parallel Genetic Algorithms are parallel stochastic algorithms applied successfully to find acceptable solutions to problems in business, engineering, and science [1]. To speed up the processing, the population is split into several sub-populations and is run in parallel. Numerous advances in this field are continuously being achieved by designing new operators, hybrid algorithms and termination criteria [4].

PGAs are not just parallel versions of sequential genetic algorithms. In fact they actually reach the ideal goal of having a parallel algorithm whose behavior is better than the sum of the separate behaviors of its component sub-algorithms. First of all, PGAs are naturally prone to parallelism since the operations on the strings are relatively independent from each other. Besides that, the whole population (panmixia) can be geographically structured [7], [8], [9] to localize competitive selection between string subsets, often leading to better algorithms. The evidences of a higher efficiency [11], [10], larger diversity maintenance [12], [13], additional availability of memory and CPU, and multi-solution capabilities [14], reinforce the importance of the research advances in the field of PGAs.

Hardware parallelization is an additional way of speeding up the execution of the algorithm, and it can be attained in many ways on a given structured-population GA. Hence, once a structured-population model is defined, it

[1]Department of CSE, PSG College of Technology, Coimbatore, India.

[2]Department of CSE, PSG College of Technology, Coimbatore, India.

[3]Department of Computer Science and Engineering, PSG College of Technology, Coimbatore, India.

E-mail : mp_psg@rediffmail.com[1]

could be implemented in any uni-processor or parallel machine. There exist many examples of this modern vision of parallel GAs, namely, a ring of panmictic GAs on a MIMD computer, a grid of individuals on uni-processor/MIMD/SIMD computers, and many hybrids.

This paper is organized as follows: Section 2 contains introductory material providing the nomenclature, and some general working principles of sequential GAs. Section 3 offers an introduction to parallel GA and formalizes the work of a parallel GA. Section 4 describes the categorization of parallel GAs. Section 5 deals with the migration issues related to parallel GAs. The communication topologies are discussed in Section 6. Some of the important applications of parallel GA are dealt in Section 7. Section 8 discusses the implementation issues. Finally, Section 9 provides some concluding remarks and summary of the survey work.

2. Sequential Genetic Algorithms

A Sequential Genetic Algorithm (SGA) [2], [3] is a heuristic used to find a vector, which is a string of free parameters with values in an admissible region for which an arbitrary quality criterion is optimized. Genetic algorithm in the beginning has randomly generated individuals, which form the population. The population in certain time is called a generation. Every individual is represented by its chromosome. The chromosomes are evaluated by its fitness function. The average fitness of the population changes gradually during the run. After a pair of individuals is chosen randomly, crossover executes an exchange of the sub-string within the pair with some probability. Mutation is an operator for a slight change of one-individual/several individuals in the population. Mutation is usually considered as a secondary search operator and its function is to restore diversity that may be lost from the repeated application of selection and crossover. Selection identifies the fittest individuals. The higher the fitness, the bigger the probability to become a parent in the next generation. Table I summarizes the meaning of these special terms in the aim of helping novel researchers.

Table I Nomenclature

Genotype	The code, devised for parameter representation of the problem in string form.
Chromosome	One encoded string of parameters (binary, Gray, Floating point type, etc.…).
Individual	One or more chromosomes with an associated fitness value.
Gene	The encoded version of a parameter of the problem to be solved.
Allele	Value that a gene can assume (binary, integer, real or complex data structures).
Locus	The position occupied by the gene in the chromosome.
Phenotype	Problem version of the genotype (algorithm version) suited for evaluation.
Fitness	Real value indicating the quality of an individual as a solution to the problem
Environment	A function representation, indicating the suitability of phenotypes.
Population	A set of individuals with their associated statistics (fitness average, hamming distance, etc).
Selection	Policy for selecting one individual from the population (selection of the fittest).
Crossover	Operation that merges the genotypes of two selected parents to yield two new children
Mutation	Operation that spontaneously changes one or more alleles of the genotype

3. Parallel Genetic Algorithms (PGAS)

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 others divide the population into several relatively isolated subpopulations. Some methods can exploit massively parallel computer architectures, while others are better suited to multicomputers with fewer and more powerful processing elements. A large population distributed among a number of semi-isolated breeding groups is known as polytypic. A PGA introduces the concept of interconnected demes. A deme is one separate population(sub-population)in many deme populations. The local selection and reproduction rules allow the species to evolve locally, and diversity is enhanced by migrations of strings among demes. Migration means an exchange rate of individuals between the demes.

3.1. Introduction to Parallel Genetic Algorithms

PGAs are a class of guided random evolutionary algorithms. A PGA has the same advantages as a serial GA, consisting in using representations of the problem parameters, robustness, easy customization for a new problem, and multiple-solution capabilities. Genetic algorithms are easily parallelized algorithms. PGA is usually faster, less prone to finding only sub-optimal solutions, and able of cooperating with other search techniques in parallel. PGAs can be divided into global, fine-grained, coarse-grained and hybrid models. A PGA can run on a network of computers or in massively parallel computers, with independence of its granularity.

There are two kinds of possible parallelism namely the data parallelism and the control parallelism. Data parallelism involves the execution of the same procedure on multiple large data subsets at the same time. Only data manipulation is parallelized and the algorithm executes one procedure in a certain period. In contrast, control parallelism involves the concurrent execution of multiple different procedures. An overview of the applications of parallel genetic algorithms (PGAs) [3], [4], [7], [10] is available. The advantages of using a PGA are illustrated in Table II. Also, there is a lot of evidence of the higher efficacy and efficiency of PGAs over traditional sequential GAs ([1], [7], [5], [8]).

Table Ii Advantages Of Using PGA

1. Parallel search from multiple points in a space
2. Works on a coding of the problem (least restrictive)
3. Basically independent of the problem (robustness)
4. Can yield alternative solutions to the problem
5. Easy parallelization as islands or neighborhoods
6. Better search, even if no parallel hardware is used
7. Higher efficiency and efficacy than sequential GAs.
8. Easy cooperation with other search procedures.

4. Classification Of Parallel GAs

There are three main types of parallel GAs : (1) global single-population master slave GAs, (2) single-population fine-grained, and (3) multiple-population coarse grained GAs.

4.1 Master slave parallelization

In a master-slave GA there is a single panmictic population (just as in a simple GA), but the evaluation of fitness is distributed among several processors. Since in this type of parallel GA, selection and crossover is considered in the entire population it is also known as global parallel GAs. This method does not affect the behavior of the algorithm. As in a serial GA, each individual may compete and mate with any other (thus selection and mating are global).

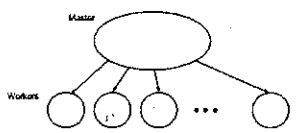

Figure 1 A schematic of a master-slave parallel GA.

As shown in Figure 1, the master stores the population, executes GA operations, and distributes individuals to the slaves. The slaves only evaluate the fitness of the individuals. The evaluation of individuals is parallelized [30] by assigning a fraction of the population to each of the processors available. If the algorithm stops and waits to receive the fitness values for all the population before proceeding into the next generation, then the algorithm is synchronous. However it is also possible to implement asynchronous master-slave GA where the algorithm does not stop to wait for any slow processors. Most global parallel GA implementations are synchronous. The global parallelization model does not assume anything about the underlying computer architecture, and it can be implemented efficiently on shared-memory and distributed memory computers. Implementation of GA was done on a shared-memory computer to search for efficient timetables for schools and they reported limited speedups [33]. In conclusion, master-slave parallel GAs are easy to implement and it can be a very efficient method of parallelization when the evaluation needs considerable computations. Besides, the method has the advantage of not altering the search behavior of the GA.

4.2 Multi-deme coarse-grain parallel GA

Multiple-population (or multiple-deme) GAs shown in Figure 2 is more sophisticated, as it contains several subpopulations, which exchange individuals occasionally. The important characteristic of multiple-deme parallel GAs are the use of a few relatively large subpopulations and migration [18]. This exchange of individuals is called migration and is controlled by several parameters. Multiple-deme GAs are very popular but also are the class of parallel GAs which is most difficult to understand, because the effects of migration are not fully understood.

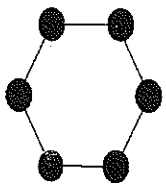

Figure 2 A schematic of a multiple-population parallel GA.

Multiple-deme parallel GAs introduces fundamental changes in the operation of the GA and has a different behavior than simple GAs. Each process is a simple GA, and there is (infrequent) communication between the populations. Multiple-deme parallel GAs is known with different names. Sometimes they are known as "distributed" GAs, because they are usually implemented on distributed memory MIMD computers[31]. Since the computation to communication ratio is usually high, they are occasionally called coarse-grained GAs. Finally, multiple-deme GAs[34] resemble the "island model" in Population Genetics which considers relatively isolated demes, so the parallel GAs are also known as "island" parallel GAs. Probably the first systematic study of parallel GAs with multiple populations was Grosso's dissertation and concluded that the favorable traits spread faster when the demes are small than when the demes are large. In a similar parallel GA approach, a copy of the best individual found in each deme is sent to all its neighbors after every generation to ensure good mixing of individuals.

4.3 Single population fine-grained parallel GA

Fine-grained parallel GAs are suitable for massively parallel computers and consists of one spatially-structured population. Selection and mating are restricted to a small neighborhood, but neighborhoods overlap

permitting some interaction among all the individuals. Figure 3 depicts the structure of this class of GAs. The ideal case is to have only one individual for every processing element available.

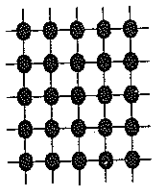

Figure 3. A schematic of a fine-grained parallel GA.

This class of parallel GAs has one spatially-distributed population, and it can be implemented very efficiently on massively parallel computers. Parallelization of the genetic algorithm of a classifier system was done on a Connection machine 1 where the parallelization was done on the selection of parents, selection of classifiers, mating and crossover [35]. A population with ladder structure [36] was used to solve some difficult combinatorial optimization problems with great success.

4.4 Hybrid parallel GA

The last method to parallelize GAs combines multiple demes with master-slave or fine-grained GAs as shown in Figure 4.This class of algorithms is called as hybrid parallel GAs, because at a higher level they are multiple-deme algorithms and with single-population parallel GAs (either master-slave or fine-grained) at the lower level. A hybrid parallel GA combines the benefits of its components, and it promises better performance than any of them alone. For example in "mixed" parallel GA algorithm, the population of each deme was placed on a 2-D grid, and the demes themselves were connected as a 2-D torus. Migration between demes occurred at regular intervals, and good results were reported for a novel neural network design and training application.

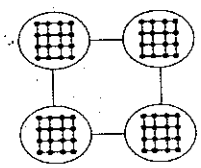

Figure 4. Hybrid GA showing a combination of a multi-deme GA (at the upper level) and a fine-grained GA (at the lower level).

Another type of hybrid parallel GA as shown in Figure 5 uses a master-slave on each of the demes of a multi-population GA. Migration occurs between demes, and the evaluation of the individuals is handled in parallel. This approach does not introduce new analytic problems, and it can be useful when working with complex applications with objective functions that need a considerable amount of computation time. Hybridizing parallel GAs shows that, a solution of the same quality of a master-slave parallel GA or a multi-deme GA can be obtained in less time[19]. Interestingly, Goldberg invented a very similar concept in the context of an object-oriented implementation of a "community model" parallel GA. In each "community" there are multiple houses where parents reproduce and the offspring are evaluated.

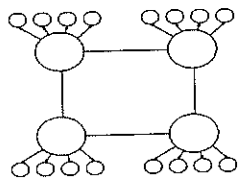

Figure 5. A schematic of a hybrid parallel GA.

At the upper level this hybrid is a multi-deme parallel GA where each node is a master-slave GA. There are multiple communities and it is possible that individuals migrate to other places. A third method of hybridizing parallel GAs is to use multi-deme GAs at both the upper and the lower levels (see Figure 6). The idea is to force panmictic mixing at the lower level by using a high migration rate and a dense topology, while a low

migration rate is used at the high level. The complexity of this hybrid would be equivalent to a multiple-population GA if the groups of panmictic subpopulations were considered as a single deme. At the lower level the migration rate is faster and the communications topology is much denser than at the upper level. Hybrid implementations can reduce the execution time more than any of their components alone.

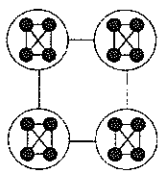

Figure 6 Hybrid structure using multi-deme GAs at both the upper and the lower levels.

5. Migration

Migration is the operator that guides the exchange of individuals among demes in a multi-deme environment. The migration of individuals from one deme to another is controlled by several parameters like (a) the topology that defines the connections between the subpopulations, (b) a migration rate that controls how many individuals can migrate and (c) a migration interval that affects the frequency of migrations.

5.1 Migration gap: In the majority of multi-deme parallel GAs, migration is synchronous which means that it occurs at predetermined constant intervals. Migration may also be asynchronous so that the demes communicate only after some events occur. An algorithm is presented where migration occurred after the demes converged completely with the purpose of restoring diversity into the demes to prevent premature convergence to a low-quality solution [37]. The same migration strategy was used later and theoretical models were presented [20],[38] which predict the quality of the solutions when a fully connected topology is used.

5.2 Migration rate: The migration rate is the parameter determining the number of individuals that undergo migration in every exchange. This value can be expressed as a percentage of the population or as an absolute value. If migration occurs too early during the run, the number of correct building blocks in the migrants may be too small to influence the search on the right direction, and expensive communication resources would be wasted. It is very common in parallel GAs that the same selection/replacement operators are used for dealing with migrants. Two alternative selection procedures for the migrants are (1) sending the best individual or (2) sending a random one.

6. Communication Topologies

A traditionally neglected component of parallel GAs is the topology of the interconnection between demes. The topology is an important factor in the performance of the parallel GA because it determines how fast a good solution disseminates to other demes. If the topology has a dense connectivity good solutions will spread fast to all the demes and may quickly take over the population. On the other hand, if the topology is sparsely connected, solutions will spread slower and the demes will be more isolated from each other, permitting the appearance of different solutions. These solutions may come together at a later time and recombine to form potentially better individuals. The communication topology is also important because it is a major factor in the cost of migration. For instance, a densely connected topology may promote a better mixing of individuals, but it also entails higher communication costs. The general trend on multi-deme parallel GAs is to use static topologies that are specified at the beginning of the run and remain unchanged. Most implementations of parallel GAs with static topologies use the native topology of the computer available to the researchers. For example,

implementations on hypercubes [6,12] and rings are common. A more recent empirical study showed that parallel GAs with dense topologies find the global solution using fewer function evaluations than GAs with sparsely connected ones [11].

The other major choice is to use a dynamic topology. In this method, a deme is not restricted to communicate with some fixed set of demes, but instead the migrants are sent to demes that meet some criteria. The motivation behind dynamic topologies is to identify the demes where migrants are likely to produce some effect. Typically, the criteria used to choose a deme as a destination includes measures of the diversity of the population [18] or a measure of the genotypic distance between the two populations [20] or some representative individual of a population, like the best.

7. Application Of Parallel GA

Many application projects using multiple-population parallel GAs has been published.The graph-partitioning problem has been a popular application of multiple-deme GAs [23]. The work on the set-partitioning [25] problem with 36,699 and 43,749 integer variables showed that the quality of the solution improved as more demes were added. Also, adequate solutions for the quadratic assignment problem [24] have been found which is another combinatorial optimization application. Coarse-grained parallel GAs have also been used to find solutions for the problem of distributing computing loads to the processing nodes of MIMD computers [31]. Another challenging application is the synthesis of VLSI circuits [29] using different types of parallel GAs to search for solutions.

8. Implementation Issues

Among all the languages for developing PGAs, *C* is the most popular. However, implementations with *C++* are now growing in number due to the advantages in software reuse, security, and extensibility. Also, because of its large capability of structured parameterization, an object oriented (OO) language has advantages in composing new prototypes. On the other hand, there exist many PGAs developed for concrete machine architectures that use embedded languages (usually versions of *C*).

Using object orientation is directly useful in developing classes for the data structures and operations contained in a parallel or sequential GA. Hence, implementing classes for genotype, population, operator pool, fitness manipulations, etc. is a natural way of coding evolutionary algorithms in general with the added advantages of any OO implementation. Communication among processes is normally achieved by using the *BSD socket interface* on UNIX systems, either directly or through the services of the well-known Parallel Virtual Machine (PVM) [21]. Some MPI and JAVA implementations are also becoming familiar. Finally, many systems simulate parallelism in a single process. The latter is useful only when the basic behavior is to be studied. However, for real-life and complex applications, a truly parallel GA is needed in order to have lower computational times and using larger populations than with sequential GAs.

9. Summary And Conclusions

This paper reviewed some of the most reviewed publications on parallel genetic algorithms. The review started with the introduction to genetic algorithms, parallel genetic algorithms and by classifying the work on this field into four categories: global master-slave parallelization, fine-grained algorithms, multiple-deme, and hybrid parallel GAs. Some of the most important contributions in each of these categories were analyzed, to try to identify the issues that affect the design and the

implementation of each class of parallel GAs on existing parallel computers.

The survey on multiple-population GAs revealed that the class of parallel GAs is very complex, and its behavior is affected by many parameters. It seems that the only way to achieve a greater understanding of parallel GAs is to study individual facets independently, and we have seen that some of the most influential publications in parallel GAs concentrate on only one aspect (migration rates, communication topology, or deme size) either ignoring or making simplifying assumptions on the others. We also reviewed publications on master-slave and fine-grained parallel GAs and realized that the combination of different parallelization strategies can result in faster algorithms. It is particularly important to consider the hybridization of parallel techniques in the light of recent results, which predict the existence of an optimal number of demes.

As GAs are applied to larger and more difficult search problems, it becomes necessary to design faster algorithms that retain the capability of finding acceptable solutions. This survey has presented numerous examples that show that parallel GAs are capable of combining speed and efficacy, and that we are reaching a better understanding which should allow us to utilize them better in the future.

References

[1] D.E.Goldberg, *"Genetic and evolutionary algorithms come of age"*, Communications of the ACM, Vol. 37, No.3, pp. 113–119, 1994.

[2] D.E.Goldberg , *"Genetic Algorithms in Search, Optimization and Machine Learning"*, Prentice Hall of India,2004.

[3] J.H. Holland , *"Adaptation in Natural and Artificial Systems"*, University of Michigan Press, Ann Arbor, Michigan, 1975.

[4] T.Bäck, D.Fogel, Z. Michalewicz, *"Handbook of Evolutionary Computation"*, Oxford University Press, London, 1997.

[5] E.Alba, *"Parallel evolutionary algorithms can achieve super-linear performance"*, Information Processing Letters, Elsevier, Vol 82, No.1, pp. 7-13, 2002.

[6 E.Alba,A.J.Nebro,J.M.Troya., *"Heterogeneous computing and parallel genetic algorithms"*, Journal of Parallel and Distributed Computing, Vol 62, No:9, pp. 1362-1385, 2002.

[7] M.Gorges-Schleuter, *"ASPARAGOS : An Asynchronous Parallel Genetic Optimisation Strategy"*, Proceedings. of the 3rd International Conference on Genetic Algorithms, pp. 422-427, Morgan Kaufmann,California,1989.

[8] E.Cantu-Paz, *"Efficient and Accurate Parallel Genetic Algorithms"*, Kluwer Academic Publishers, Boston, 2000.

[9] E.Alba, J.M.Troya, *"Improving flexibility and efficiency by adding parallelism to genetic algorithms"*, Statistics and Computing, Vol 12 (2) : 91-114, 2002.

[10] J. Stender , *"Parallel Genetic Algorithms: Theory and Applications"* IOS Press, Amsterdam, 1993.

[11] V. S. Gordon, D. Whitley, *"Serial and Parallel Genetic Algorithms as Function Optimizers"*, Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann, California, pp. 177-183, 1993.

[12] S.Cahon, E.G.Talbi, N.Melab, *"ParadisEO: A Framework for the flexible design of parallel and distributed hybrid metaheuristics"*, Journal of Heuristics, Vol. 10, No.3, pp. 357-380,Kluwer Academic Publishers, May 2004.

[13] H. Mühlenbein. *"Evolution in Time and Space - The Parallel Genetic Algorithm"*, Foundations of Genetic

Algorithms, pp. 316-337, Morgan Kaufmann, California, 1991.

[14] T. C. Belding. *"The Distributed Genetic Algorithm Revisited"*, Proceedings of the 6th International Conference on Genetic Algorithms, pp. 122-129, Morgan Kaufmann, California, 1995.

[15] Adamidis. P., *"Review of parallel genetic algorithms bibliography"*, Technical Report, Aristotle University of Thessaloniki, Greece, 1994.

[16] D.E.Goldberg , Deb K., Clark J. H., *"Genetic algorithms, noise, and the sizing of populations"*, Complex Systems, vol. 6,No.4, pp. 333–362, 1992.

[17] G.Harik , Cantu-Paz E., Goldberg D. E., Miller B. L., *"The gambler's ruin problem, genetic algorithms, and the sizing of populations"*, In Proceedings of IEEE International Conference on Evolutionary Computation, pp. 7–12, IEEE Press , New Jersey, 1997.

[18] S.C Lin, Punch W., Goodman E., *"Coarse-Grain Parallel Genetic Algorithms : Categorization and New Approach"*, In Sixth IEEE Symposium on Parallel and Distributed Processing, IEEE Computer Society Press, Los Alamitos, California, October 1994.

[19] R. Bianchini, Brown C. M., *"Parallel genetic algorithms on distributed-memory architectures"*, Transputer Research and Applications, IOS Press, Amsterdam, pp. 67–82, 1993

[20] M. Munetomo, Takai Y., Sato Y., *"An efficient migration scheme for sub population- based asynchronously parallel genetic algorithms"*,. In Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California, page 649, 1993.

[21] Marin F. J., Trelles-Salazar O., Sandoval F., *"Genetic algorithms on LAN-message passing architectures using PVM: Application to the routing problem"*, Parallel Problem Solving from Nature, , Springer-Verlag ,Berlin, pp. 534–543, 1994.

[22] Cantu-Paz E., Mejia-Olvera M., *"Experimental Results in Distributed Genetic Algorithms"*, In International Symposium on Applied Corporate Computing, pp. 99– 108, Monterrey, Mexico, 1994.

[23] Talbi E.G, Bessiere P., *"A Parallel Genetic Algorithm for the Graph Partitioning Problem"*, In Proceedings of the International Conference on Supercomputing, Cologne, France, June 1991.

[24] Li T., Mashford J., *"A parallel genetic algorithm for quadratic assignment"* ,In Proceedings of the ISMM International Conference., pp. 391–394, Acta Press, Anaheim, California, 1990.

[25] D. Levine, *"A Parallel Genetic Algorithm fot the Set Partitioning Problem". T. R. No. ANL-94/23*, Argonne National Laboratory, Mathematics and Computer Science Division, 1994.

[26] Seredynskif, *"Dynamic mapping and load balancing with parallel genetic algorithms"*, In Proceedings of the First IEEE Conference on Evolutionary Computation, vol. 2, pp. 834–839, IEEE Press, Piscataway, NewJersey, 1994.

[27] B. Manderick, P. Spiessens, *"Fine-Grained Parallel Genetic Algorithms"*, Proceedings of the 3rd International Conference on Genetic Algorithms, George Mason University, USA, pp. 428- 433,1989.

[28] M. Mejía-Olvera, E. Cantú-Paz, *"DGENESIS-Software for the Execution of Distributed Genetic Algorithms"*, Proceedings of the XX Conferencia Latinoamericana de Informática, pp. 935-946, Monterrey, México, 1994.

[29] Davis M., Liu L., Elias J. G, *"VLSI circuit synthesis using a parallel genetic algorithm"*, In Proceedings of the First IEEE Conference on Evolutionary

Computation, pp. 104–109, IEEE Press, Piscataway, New Jersey, 1994.

[30] Abramson D., Abela J., *"A parallel genetic algorithm for solving the school timetabling problem"*, In Proceedings of the Fifteenth Australian Computer Science Conference, vol. 14, pp. 1–11, Australia, 1992.

[31] Hauser R., Manner R., *"Implementation of standard genetic algorithm on MIMD machines"*, pp. 504–513, Springer-Verlag , Berlin, 1994.

[32] J. C. Potts, T. D. Giddens, S. B. Yadav. *"The Development and Evaluation of an Improved Genetic Algorithm Based on Migration and Artificial Selection"*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 24, No.1, pp. 73-86, January 1994.

[33] Cantu-Paz E, *"Designing efficient master-slave parallel genetic algorithms"*, Illinois Genetic Algorithms Laboratory, Urbana, Illinois, 1997.

[34] Cohoon J. P., Martin W. N., Richards D. S., *"A multi-population genetic algorithm for solving the K-partition problem on hyper-cubes"*, In Proceedings of the Fourth International Conference on Genetic Algorithms, California pp. 244–248, 1991.

[35] Robertson G., *"Parallel implementation of genetic algorithms in a classifier system"*, In Proceedings of the Second International Conference on Genetic Algorithms, pp. 140–147, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1987.

[36] Gorges-Schleuter M., *"Asparagos : A population genetics approach to genetic algorithms"*, In Evolution and Optimization , pp. 86–94., Akademie-Verlag, Berlin, 1989.

[37] Braun H. C., *"On Solving Travelling salesman Problems by Genetic Algorithms"*, Parallel Problem Solving from Nature, pp. 129–133, Springer-Verlag, Berlin, 1990.

[38] Cantu-Paz E., Goldberg D. E., *"Modeling Idealized Bounding Cases of Parallel Genetic Algorithms"*, Second Annual Conference, Morgan Kaufmann, California, 1997.

Author's Biography

Ms P Visalakshi received the ME degree in Applied Electronics from PSG College of Technology, Bharathiar University. She is Currently working as a Senior Lecturer, Department of CSE, PSGCT. Her research interest includes Evolutionary Computing, Neural Networks, Operating Systems, Parallel and Distributed Computing.

Ms T Hamsapriya received the ME degree in Communication Systems from PSG College of Technology , Bharathiar University. She is currently working as an Assistant Professor, Department of CSE, PSGCT. Her research interest includes Parallel and Distributed Computing, Evolutionary Computing, Neural Networks and Data Mining.

Dr S N Sivanandam, received the PhD degree in Electrical and Electronics Engineering from Madras University, Chennai in 1982. He is currently serving as the Professor and Head of the Computer Science and Engineering Department, PSGCT, Coimbatore. His research interest lies in the area of Control Systems, Neural Networks, Genetic Algorithm, Digital Logic Design. He has published 400 Technical papers in International, National Journals and Conferences. He has published 7 books.