MoHPBGA: Multi-objective Hierarchical Population Balanced Genetic Algorithm using MapReduce

Poka Laxmi
Research Scholar
Department of Computer Science, Vishwakarma Institute of Technology, Pune, India

Jayant Umale
Professor
Department of Computer Science, Vishwakarma Institute of Technology, Pune, India

Sunita Mahajan
Principal
Institute of Computer Science
Mumbai Education Trust, Bandra, Mumbai, India

ABSTRACT

Use of heuristic methods is common to find the solutions to the optimization problems for scientific and real time. Problems such as Travelling Salesman (TSP) require more accurate solution which is tried by various optimization methods. Research in this direction shows the use of Genetic algorithms (GA) as promising candidate and is preferred over other optimization methods. Firstly due to the use of large population and secondly large number of iterations GA tends to be more accurate but inefficient with respect to computation time. Variants of GA are formulated and experimented so as to take care of execution time. We present the review of approaches used to formulation of GA solutions mainly parallel GA (PGA), distributed GA (DGA) and hierarchical parallel GA (HPGA). Further this paper proposes Multi objective Hierarchical Population Balanced Genetic Algorithm (MoHPBGA) as the improved candidate which uses map reduce framework for efficient use of population mapping and synchronization of tasks.

General Terms

Multiobjective GA, Mapreduce, Hierarchical Parallel GA, Popolation balancing in GA.

Keywords

Optimization, Genetic Algorithms, Parallel Computing, Multiobjective problems.

1. INTRODUCTION

Genetic algorithms are stochastic search methods introduced by J Holland in the 1970's and inspired by the biological evolution of living beings. So these algorithms belongs to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, selection, crossover and mutation. The new generation evolves from the population of existing individuals and continues to improve over generations by eliminating weak individuals and using best few for further generations. The participation of the individual in generation process depends on its fitness. Fitness is the criteria based on the goal of required improvement for example lower cost of travelling time for the distances in shortest distance problem derived from parameters affecting distance properties. Multiple individuals stochastically selected based on their fitness from the current population to participate in crossover (recombination) and mutation (randomly changing selected parameter value) to form a new individuals. Process continues iteratively to produce further generations and terminates when either desired number of generations has been produced or a sufficient fitness level has been reached with the population. An abstraction of a typical GA is given as below [1] in the algorithm by the Fig. 1.

```
Algorithm 1: Algorithm for Generic Genetic Algorithm
Generate initial population, G(0);
Evaluate G(0);      (apply fitness function)
t=1;
Repeat
Generate G(t) using G(t-1);   (apply operators)
Evaluate (decode (G(t)));
t=t+1;
Until solution is found or termination.
```

Fig 1: Algorithm for Generic Genetic Algorithm

1.1 Sequential GA Issues

As the GA executes sequentially by steps, as crossover and mutation performed on pair of individuals and for large population in consideration for these steps, the time required to compute is also higher. Number of objectives to be satisfied is the yet another factor that further delays the execution. Let m is the number of objectives and n is the number of individuals taking part in process then it amounts to O(mn)3, which means to get more number of iterations for accuracy of solution executing on large population will lead to very high execution time of the order of cubical time.

Another issue with sequential implementation is its memory utilization for storing large population. The run time memory requirements are also higher since the population must be available in memory for the operations of GA. Although GA normally leads to optimal solution but at some instances sequential GAs may get trapped in a sub-optimal region of the search space thus becoming unable to find better quality solutions.

Because of above mentioned issues the problems which require quicker solutions resist the use of genetic algorithms for computation. This is obvious because the genetic algorithm does not find solutions using mathematical representation of the problem, but it is more or less a stochastic, discrete, non-linear, and a high-dimensional seeking algorithm. The use of genetic algorithms can be promoted by reduction of execution time to find quicker solution. So, we summarize the efforts towards the various implementations of GA in next section

2. RELATED WORK

GA was developed by John Holland (1975) over the course of the 1960s and 1970s and finally popularized by one of his students, David Goldberg, who was able to solve a difficult problem involving the control of gas-pipeline transmission for his dissertation (Goldberg, 1989). Holland’s original work was summarized in his book. He was the first to try to develop a theoretical basis for GAs through his schema theorem. The work of De Jong (1975) showed the usefulness of the GA for function optimization and made the first concerted effort to find optimized GA parameters. Goldberg has probably contributed the most fuel to the GA fire with his successful applications and excellent book (1989). Since then, many versions of evolutionary programming have been tried with varying degrees of success.

In the first International Conference of Genetic Algorithms (ICGA), there were no papers about parallel GAs at all. This situation changed in the second ICGA in 1987, where six papers were published. From then on there has been a steady flow of papers published in conferences and journals of GAs and parallel computation. To reduce the computation load of genetic search, a lot of methods for searching in parallel and distributed manner have been proposed [9] [10] [29]. The number of papers, dissertations and books on the theory of these genetic algorithms has been increasing. Some of them are briefly reviewed in the following.

In 1992, D. Abramson and J. Abela presented a paper in which they discussed the application of a GA to the school timetabling problem [3], and showed that the execution time can be reduced by using a commercial shared memory multiprocessor. The program code was written in Pascal and run on an Encore Multimax shared memory multiprocessor. The times were taken for a fixed number of generations (100) so that the effects of relative solution quality could be ignored and proved that the speedup was attained from a parallel implementation of the program.

A PGA developed by Suh and Gucht [17], has been applied to TSPs of growing size (100-1000 cities) problems. PGAs without selection and crossover, so called independent strategies were used. These algorithms consist of running an unlimited number of independent sequential local searches in parallel. PGAs with low amount of local improvement were used and performed better in terms of quality solution than the independent strategies. In terms of computational time, the PGA showed nearly a linear-speedup for various TSPs, using up to 90 processors. The algorithms were run on a BBN Butterfly.

In 1998, Ranieri and Raffaele [25] discussed the results of the application of parallel GA algorithms to the TSP. Both fine grained and coarse grained parallel GAs which adopt the selected genetic operators were designed and implemented on a 128-node nCUBE 2 multicomputer. The tests showed that the two point crossover finds better solutions, as does replacement criteria. They used an innovative mapping strategy for fine grained algorithm that makes the number of solutions managed independent of the number of processing nodes used. For the coarse grained GA it was observed that the quality of solutions gets worse if the number of nodes used were increased. Moreover, due to the sorting algorithm used to order each sub population by fitness, the speedup of the coarse grained GA was superlinear. Whereas, for fine grained GA the quality of solutions does not get worse if the number of the nodes used is increased, they showed good scalability. A comparison between the fine and coarse grained algorithms highlighted that fine grained algorithms represent the better compromise between quality of the solution reached and the execution time spent on finding it. Complete performance results showing the behavior of Parallel Genetic Algorithms for different population sizes, number of processors used, migration strategies were reported.

In [31], the authors have very well presented a comparative analysis of five different types of coarse grained PGAs. Four conceptually different PGA approaches and one hybrid of two of these approaches were compared using the TSP as an example application problem. For fair comparisons, all PGAs were developed based on the same baseline SGA, implemented on the same 16 thin PEs of an IBM SP2 parallel machine, and tested using the same set of initial populations on the same set of TSP instances. As a result of the experiments conducted in this study, a particular PGA that combines a new sub tour technique with a known migration approach was identified to be the best. The results showed that the migration and the segmentation-migration approaches were able to find solutions of similar high quality faster than the other approaches.

Also one of the very fruitful studies was the dissertation of Cant´u-Paz (2000) [8]. The dissertation brought many new principles of the rational design of fast and accurate parallel genetic algorithms. It helped many researchers to decide a configuration of the many options of topologies, migration rates, number and size of demes. The important findings were brought to light as importance of accurate population sizing for PGA, an equivalent scalability of single and multiple demes, impracticability of isolated demes, improvement quality and efficiency by migration, advantage of fully connected topologies, studies of effects of topology and optimal allocation computing resources.

The article [32] gives a brief overview of theoretical advances, computing trends, applications and future perspectives in parallel genetic algorithms. It is basically a survey of past and recent developments in parallel genetic algorithms. The information is segregated into two periods before and after the year 2000 and in all chapters. The relevant issues connected with parallel genetic algorithms were highlighted. The survey hinted several views, whose range from new genetic theories over parallel computing and wide and various ranges of real-world applications to future developments, challenges and perspectives for parallel (genetic) metaheuristcs.

Apart from using PGAs for solving various problems, people have also worked on implementing these PGAs on various different platforms using different technologies. Some of such works is described below and also presented in a tabulated form in Table 1.

In 2004, the author [22] has presented his work by using a master-slave paradigm on a Beowulf Linux cluster using MPI programming library. With the help of this he has written a pseudo code that first initializes the basic population and also the number of nodes present in the cluster. Then it assigns the fitness function to the slave or other nodes in the cluster. The slave ones compute the fitness objective and do the mutation. After this, they send back the result to the master where it makes the final counterpart. This paper presented a view of implementation and realization of algorithms on a parallel architecture.

As MPI’s are associated with some drawbacks of unable to handle heterogeneity, failures etc and that is why it not suitable for cloud applications, another author [2] presented his work of executing PGAs on a MapReduce model. This model provides a parallel design pattern for simplifying application developments in distributed environments. As this model cannot be used to express PGA’s directly, they presented an extension to MapReduce model through an additional reduce phase for global selection, called once at the end of each iteration of the GA loop. To simplify handling faults during execution, they made the master to replicate the optimum individuals selected by MRPGA for each round of evolvement in their architecture.

Table 1. Implementation of PGAs On Different Platforms

Parame-ters / Approaches	Technology /platform	Type of PGA	Comments on Performance
PGA using MPI [22]	Beowulf Linux Cluster and MPI.	GPGA	Homogenous cluster. Parallel implementation and Performance improvement.
PGA using CUDA[24]	intel Core i7 and nVidia graphics cards: 8800 GTX, GTX 285 and GTX 260-SP216.	CPGA	Improved Performance for increasing population size. Slows for smaller no. of islands and lower population size.
PGA using MapReduce [2]	Aneka (.Net based enterprise Grid software platform) and C#	CPGA	Handles heterogeneity and supports MapReduce.

Parallel genetic algorithms are usually implemented on parallel machines or distributed systems. But Petr Pospichal, Jiri Jaros, and Josef Schwarz [24] provided with a new view of implementation of parallel genetic algorithms to programmable graphics hardware found in commodity PC. Their paper deals with the mapping of the parallel island based genetic algorithm with unidirectional ring migrations to nVidia CUDA software model. The proposed mapping was tested using Rosenbrock's, Griewank's and Michalewicz's benchmark functions. Their results indicate that this approach leads to speedups up to seven thousand times higher compared to one CPU thread while maintaining a reasonable results quality, which clearly shows that GPUs have a potential for acceleration of GAs and allow solving much complex tasks.

Although much work is reported about PGA models (and implementations on different parallel architectures), involving SOPs (single objective optimization problems), PGA models could also be applied for multiobjective optimization problems (MOPs) [5]. MOPs normally have several (usually conflicting) objectives that must be satisfied at the same time. The first multi-objective GA, called Vector Evaluated Genetic Algorithms (or VEGA), was proposed by Schaffer [27]. Afterward, several major multi-objective evolutionary algorithms were developed such as Multi-objective Genetic Algorithm (MOGA) [12], Niched Pareto Genetic Algorithm [15], Random Weighted Genetic Algorithm (RWGA)[23], Nondominated Sorting Genetic Algorithm (NSGA) [28], Strength Pareto Evolutionary Algorithm (SPEA) [33], Pareto-Archived Evolution Strategy (PAES) [19], Fast Non-dominated Sorting Genetic Algorithm (NSGA-II) [6], Multi-objective Evolutionary Algorithm (MEA) [26], Rank-Density Based Genetic Algorithm (RDGA) [20]. Although there are many variations of multiobjective GA in the literature, these cited GA are well-known and credible algorithms that have been used in many applications and their performances were tested in several comparative studies.

Apart from implementing GA by parallelization, researchers have tried to implement them on distributed systems. For e.g. in [29] the authors proposed a new distributed GA with a distributed environment scheme. In this scheme, a whole population is divided into several subpopulations, and the GA parameters such as the mutation rate and the crossover rate in each subpopulation were different from each other. The migration operation was performed similarly as the conventional distributed GA. They demonstrated the effectiveness of the proposed scheme with 9 subpopulations on a parallel computer, nCUBE2, with 64 processors and one processor assigned to one population. The problem solved was the maximization of the Rastrigin function. The results were then compared with SPGA (single population GA) and MPGA (multiple population GA), which showed a clear performance gain in DEGA, which can be attributed to the reason of various environments. Thus, they concluded that the distributed environment GA is the fastest way to gain the good solution under the given population size and uncertainty of the appropriate crossover and mutation rates.

A similar kind of proof was given by Yiyuan Gong and Alex Fukunaga [14], by proposing the use of an extremely simple, randomized strategy for setting control parameters in a distributed genetic algorithms, where a different, random set of control parameter values (mutation rate, crossover rate, population) were used on every processor in an island-model distributed GA. This randomized, heterogeneous distributed GA exploits the fact that, sufficiently sampling the space of control parameter spaces can result in a near-optimal set of control parameters being discovered (for a single processor). Apart from proving that these Heterogeneous DGAs are a good option for solving problems, the results also showed that as the number of processors were increased, the relative performance of the heterogeneous DGA improves and this is because of the fact that with a sufficient number of processors, it is likely that at least some of the processors will end up being assigned a set of random control parameters which performs particularly well on a given problem.

A totally different variant was proposed in [13], where the author proposed a new architecture for distributed GAs, based on distributed storage of the individuals in a persistent pool. This approach was tailored for distributed systems in which processors are loosely coupled, failure-prone and can run at different speeds. They simulated the pool GA approach on some of application problems which are continuous functions and also to a real-world product lifecycle design problem, using crossover and mutation operators and the results showed clear advantage of using concurrent processing. They also simulated crashes during execution of Pool GA. The results indicated that the algorithm is tolerant up to half the processors crashing at random times without significantly affecting the performance.

Several hierarchical algorithms have also been suggested in the recent past, e.g. in 1997; Herrera, Lozano and Moraga discussed a paper on hierarchical distributed genetic algorithms (HDGA) [11]. In this they presented a hierarchical model of distributed genetic algorithm in which a higher level distributed genetic algorithm joins simple distributed genetic algorithms. Also with the union of hierarchical structure presented and the idea of heterogeneous distributed genetic algorithms, they proposed a heterogeneous hierarchical distributed genetic algorithm and showed that the proposed model consistently outperforms equivalent sequential genetic algorithms and simple distributed genetic algorithms. This paper was the first investigation on a basic implementation of this idea.

Another variant of hierarchical algorithms is included in [21]. In this HGAs are explained in details and the advantages of a multi-layered hierarchical topology are clearly shown. The experiment was carried out in the field of computational fluid dynamics (CFD) shape optimization. The optimization task consists of reconstructing the shape of a converging-diverging nozzle for a transonic flow involving a shock. The optimization was presented by a real coded GA, with a single population of size 20, then again tested using HGA (coarse grained, so we can say it as DGA) with a grid of size 100 and then finally tested for HGA with multiple layers i.e. grid of size 100 at layer 1, 50 grid points at layer 2 and grid of size 25 at the bottom layer. They summed up the performances for traditional Gas, hierarchical GAs (DGAs) and hierarchical Gas with multiple models. They focused on the CPU time needed by each of these 3 approaches and experiment and showed that the hierarchical GAs are on an average 3 times faster than either of the other approaches. Therefore they concluded that GAs can handle well approximate models within a hierarchical topology and thus these models of different complexities can significantly speed-up an optimization process.

Also in 2006, Lim, Y. Ong, Y. Jin, B. Sendhoff, and B. Lee, proposed a Hierarchical Parallel Genetic Algorithm framework (GE-HPGA) [4] based on standard grid technologies. The framework offers a comprehensive solution for efficient parallel evolutionary design of problems with computationally expensive fitness functions, by providing novel features that conceals the complexity of a Grid environment through an extended GridRPC API and a metascheduler for automatic resource discovery. To assess the effectiveness of the framework, theoretical analysis on maximum speed-up of GE-HPGA and the practical conditions that must be fulfilled for any speed-up has been reported

3. PROPOSED WORK

3.1 Multiobjective Hierarchical Population Balanced GA using MapReduce (MoHPBGA)

It is very evident from the literature that various variants of GAs, especially hierarchical GAs (HGAs) [11][21][4] are better performers than the sequential ones. So we take into account this form of GA, and propose how these HGAs can be applied to a multiobjective optimization problem like Travelling Tournament Problem (TTP) [18], which is a sports timetabling problem that abstracts the important issues in creating timetables where team travel is an important issue using MapReduce [16][30], which is a software framework that is used to support distributed computing on large data sets on a clusters of computers. So in the following section we present an architecture showing how a multiobjective problem can be solved on this framework in a hierarchical way.

3.2 Proposed Architecture

As shown in Fig. 2, we propose to decompose GA using both task (objective) and data (population) strategies. The structure for decomposition will be hierarchical and all the local clusters will offered the same set of population, which then will be working on this population but each with a different objective. The global level of decomposition will work by decomposition of multiple GA tasks (objectives) which are assigned to local decomposition level and the local level will be responsible to decompose the offered population and run parallel tasks on the local infrastructure- computing cluster. The diagram shows the local and global level decomposition of GA.

We propose to use MapReduce model of programming. The MapReduce model provides a parallel design pattern for simplifying application developments in distributed environments. In MapReduce we can split a large problem space into small pieces and parallelize the execution using small tasks on the smaller space. This was proposed by Google for easily harnessing a large number of resources in data centers to process data-intensive applications. Hadoop allows users to deligate tasks to the middleware and distribute as per given strategy. This happens more transparently as hadoop supports this by the implementation of low level communication interfaces for cluster.

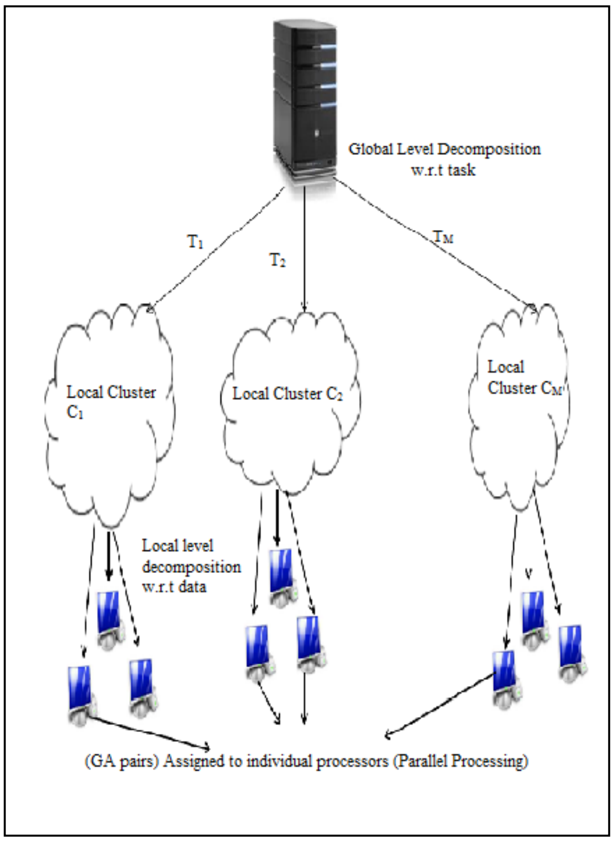
Global Level Decomposition w.r.t task
T_1
T_2
T_M
Local Cluster C_1
Local Cluster C_2
Local Cluster C_M
Local level decomposition w.r.t data
(GA pairs) Assigned to individual processors (Parallel Processing)

Fig 2: Proposed System Architecture

3.3 Algorithm and Flowcharts

We propose the following algorithm for the above mentioned architecture.

Algorithm 2: Multiobjective Hierarchical Population Balanced GA using MapReduce (MoHPBGA)

```
1. Accept input (objectives) from the user.
2. Generate at random a population, P, of chromosomes i.e. the solution space.
3. Decompose the problem based on objectives i.e. assignment of one objective to one cluster (done by server machine).
4. Parallel execution on each cluster, on given population based on objective.
5. On an individual cluster:
    5.1 Accept the population (P) and the objective (O).
    5.2 Perform PGA on cluster.
        5.2.1 Divide P into $SP_1$, $SP_{2....}SP_N$ subpopulations.
        5.2.2 For SPi, i=1... N, execute in parallel the next steps on the available nodes (N).
            5.2.2.1 Apply the selection mechanism and the genetic operators using MapReduce.
            5.2.2.2 If the stop criteria not fulfilled, return to 5.2.2
    5.3 Combine the results from the all the nodes.
    5.4 Send the best results to the top i.e. server machine.
6. Gathering of best results from all clusters on server machine.
7. Run a Local GA on server to get the best results satisfying multiple objectives.
8. Display the Best or optimized final results.
```

Fig 3: Proposed MoHPBGA Algorithm

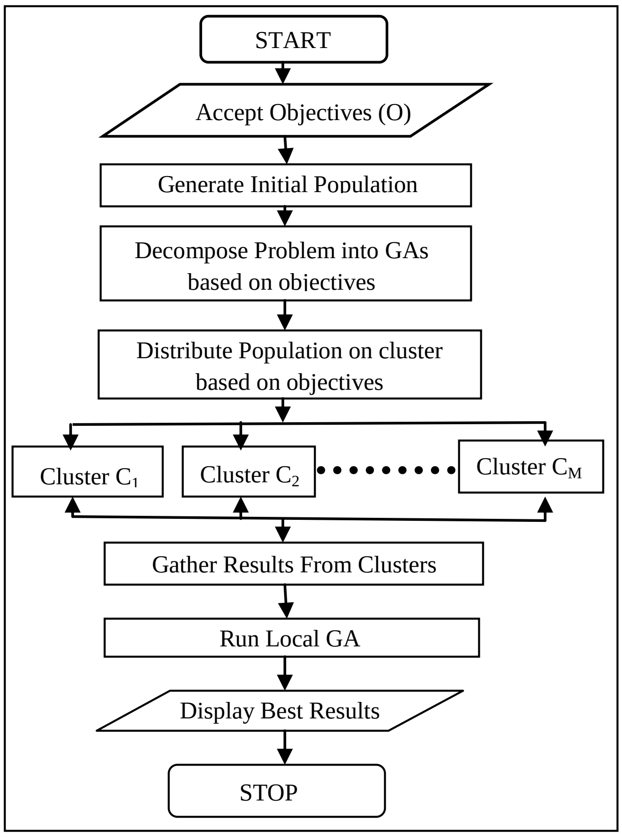
START
Accept Objectives (O)
Generate Initial Population
Decompose Problem into GAs based on objectives
Distribute Population on cluster based on objectives
Cluster C_1
Cluster C_2
Cluster C_M
Gather Results From Clusters
Run Local GA
Display Best Results
STOP

Fig 4: Flowchart showing flow of operation on server side

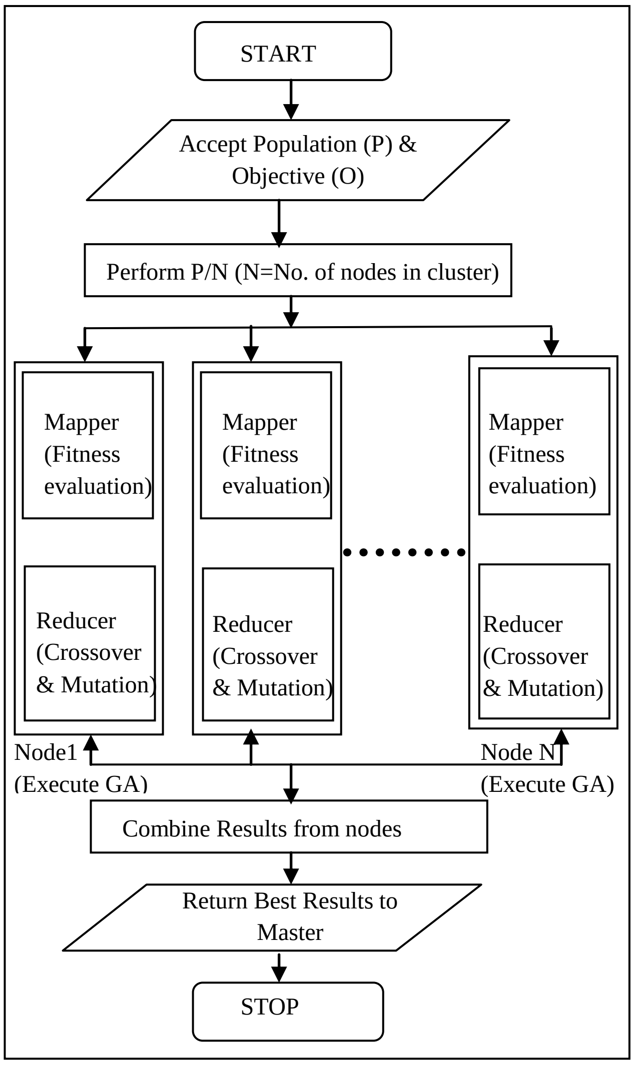
START
Accept Population (P) & Objective (O)
Perform P/N (N=No. of nodes in cluster)
Mapper (Fitness evaluation)
Reducer (Crossover & Mutation)
Node1 (Execute GA)
Node N (Execute GA)
Combine Results from nodes
Return Best Results to Master
STOP

Fig 5: Flowchart showing flow of operations on each individual cluster

In our approach hierarchical decomposition takes place by first decomposing the problem based on number of objectives i.e. each objective Ki executes on individual clusters Ci and at the second stage population balancing is done by parallel decomposition of the offered population. If suppose P is the offered population size to each cluster & N is the number of nodes in the cluster, then each node gets a P/N number of individuals for genetic evaluations. Then the results from all the participating nodes in a cluster are combined and the best results satisfying the given objective (Ki) is send to the higher level for execution of multiobjective GA on server. Thus we get an n-dimensional vector x, containing the individuals satisfying an objective function individually. Now at this stage on server, a final local GA is carried out on x to find x*, satisfying a set of objective functions.

The algorithm and the flowcharts present the operations that would be carried out to implement MoHPBGA. For multiobjective optimization, given an n-dimensional decision variable vector x={x1, x2 ...xn} in the solution space P, we need to find a vector x* that either minimizes or maximizes a given set of K objective functions (O) z(x*)={z1(x*), z2(x*)... zk(x*)}

4. CONCLUSION AND FUTURE WORK

Genetic Algorithms are easy to apply to a wide range of problems, from optimization problems like the travelling salesperson problem, to inductive concept learning, scheduling, and layout problems. The results can be very good on some problems, and rather poor on others. Implementing these GAs with modification have proven to be very useful, especially hierarchical GAs, resulting in faster and more robust algorithms.

MoHPBGA presented in this paper is a genetic algorithm designed for hierarchical classification for multiobjective problems. Our proposed method executes the instances of GA on the clusters and uses the complete population for optimization. Further the instance of GA on each cluster executes using PGA on partitioning population given to the cluster. Compared to other algorithms in the literature, MoHPBGA has the advantage of dealing with both local and global information simultaneously. Also, the hypothesis shows the improvement of computation time compared to earlier HPGA approaches.

As future work, we implement this concept of MoHPBGA on Hadoop platform, which inherently supports for MapReduce programming model along with the the performance analysis of the system.

5. ACKNOWLEDGEMENT

I would like to acknowledge and extend my heartfelt gratitude to the all the persons who have made the completion of this paper possible. l owe my deep regards and honor to express my gratitude to Mr. Bhushan Thakare, Manish Chandak, Miss. Reshu Prowal, Madhuri Aher and all the faculty members of VIT college for providing valuable support, guidance, help and inspiration all through this paper.

6. REFERENCES

[1] Bart Ian Rylander, “Computational Complexity and the Genetic Algorithm”, Doctoral Dissertation, University of Idaho Moscow, ID, USA @2001, ISBN: 0-493-33514-5.

[2] Chao Jin, Christian Vecchiola and Rajkumar Buyya, “MRPGA: An Extension of MapReduce for Parallelizing Genetic Algorithms”, in Proceedings of the 4th IEEE International Conference on e-Science 2008.

[3] D. Abramson and j. Abela, “A parallel Genetic Algorithm for Solving the School Timetabling Problem”, in Proceedings of the 15th Australian Computer Science Conference, Hobart, Feb 1992, pp 1-11.

[4] D. Lim, Y. Ong, Y. Jin, B. Sendhoff, and B. Lee, “Efficient Hierarchical Parallel Genetic Algorithms using Grid Computing”, in Proceedings of the Future Generation Computer System, 23(4):658–670, 2007.

[5] De Toro, F.; Ortega, J.; Fernandez, J.; Diaz, A.; “PSFGA: A Parallel Genetic algorithm for Multiobjective Optimization”, in Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, 2002.

[6] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., “A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II”, IEEE Transactions on Evolutionary Computation 6(2) : 182-197, 2002.

[7] E.Alba and J.M.Troya, “A Survey of Parallel Distributed Genetic Algorithms”, Complexity 4(4), 31-52, 1999. John Wiley and Sons, Inc.

[8] E.Cant´u-Paz, “Efficient and Accurate Parallel Genetic Algorithms”, Kluwer Academic Publishers, Norwell, MA, USA, 2000, ISBN: 978-0-7923-7221-9.

[9] Erick Cant`u-Paz, “A Survey of Parallel Genetic Algorithms”, Calculateurs Paralleles, Reseaux et Systems Repartis, 10(2):141-171, 1998.

[10] Erick Cant`u-Paz, David E. Goldberg, “Are Multiple Runs of Genetic Algorithms Better than One?”, in Proceedings of the 2003 International Conference on Genetic and Evolutionary Computation (GECCO ’03).

[11] F.Herrera, M. Lozano, M. Lozano, “Hierarchical Distributed Genetic Algorithms”, Technical Report #DECSAI-97-01-03, Dept. Of Computer Science and Artificial Intelligence, University of Granada, Spain, 1997.

[12] Fonseca, C.M. and Fleming, P.J, “Multiobjective Genetic Algorithms”, in IEE Colloquium on Genetic Algorithms for Control Systems Engineering (Digest No. 1993/130), 1993.

[13] Gautam Roy, et al, “A Distributed Pool Architecture for Genetic Algorithms”, in Proceedings of the 2009 IEEE Conference on Evolutionary Computation (CEC 2009).

[14] Gong, Y., Fukunaga, A, “Distributed Island-Model Genetic Algorithms Using Heterogeneous Parameter Settings”, in Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2011.

[15] Horn, J., Nafpliotis, N., and Goldberg, D.E, “A Niched Pareto Genetic Algorithm for Multiobjective Optimization”, in Proceedings of the 1st IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, 1994.

[16] J. Dean and S. Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters”, in Proceedings of the 6th Conference on Symposium on Systems Design and Implementation,OSDI ’04.

[17] J. Suh and D. Van Gucht, “Distributed Genetic Algorithms”, Tech. Report 225, Computer Science Department, Indiana University, Bloomington, 1987.

[18] Kelly Easton, Nemhauser George L., and Trick Michael A, “The Traveling Tournament Problem Description and Benchmarks”, in Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming, CP’01.

[19] Knowles, J.D. and Corne, D.W., “Approximating the nondominated front using the Pareto archived evolution strategy”, Evolutionary Computation, 8(2):149-172, 2006.

[20] Lu, H. and Yen, G.G., “Rank-density-based multiobjective genetic algorithm and benchmark test function study”, IEEE Transactions on Evolutionary Computation, 7(4):325-343, 2003.

[21] M. Sefrioui, K. Srinivas and J. Periaux, “Aerodyanmic Shape Optimization using a Hierarchical Genetic Algorithm”, European Conference on Computational Methods in Applied Science and Engineering (ECCOMAS 2000).

[22] Muhammad Ali Ismail, “Parallel Genetic Algorithms (pgas): master slave paradigm approach using mpi”, E-Tech 2004.

[23] Murata, T. and Ishibuchi, H, “MOGA: multi-objective genetic algorithms”, in Proceedings of the 1995 IEEE International Conference on Evolutionary Computation, Perth.

[24] Petr Pospichal, Jiri Jaros, and Josef Schwarz, “Parallel Genetic Algorithm on the CUDA Architecture”, in Proceedings of the Applications of Evolutionary Computation, 6024 : 442-451, 2010.

[25] Ranieri Baraglia, Raffaele Perego, “Parallel Genetic Algorithms for Hypercube Machines”, in Proceedings of the 3rd international conference on vector and parallel processing VECPAR’98.

[26] Sarker, R., Liang, K.-H., and Newton, C., “A new multiobjective evolutionary algorithm”, European Journal of Operational Research 140(1):12-23, 2002.

[27] Schaffer, J.D, “Multiple Objective optimization with vector evaluated genetic algorithms”, in Proceedings of the International Conference on Genetic Algorithm and their applications. 1985.

[28] Srinivas, N. and Deb, K., “Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms”, Journal of Evolutionary Computation, 2(3): 221-248, 1994.

[29] T. Hiroyasu, M. Kaneko, K. Hatanaka, “A parallel genetic algorithm with distributed environment scheme”, in Proceedings of the 1999 IEEE International conference on Systems, Man, and Cybernetics (SMC '99).

[30] Verma, A. Llora, X. Goldberg, D.E. Campbell, R.H, “Scaling Genetic Algorithms Using MapReduce”, in Proceedings of the 9th International Conference on Intelligent Systems Design and Applications, ISDA '09.

[31] Wang, L.; Maciejewski, A.A.; Siegel, H.J.; Roychowdhury, V.P, “A comparative study of five parallel genetic algorithms using the traveling salesman problem”, in Proceedings of the 1st Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing, 1998.

[32] Zdenek Konfrst, “Parallel Genetic Algorithms: Advances, Computing Trends, Applications and Perspectives”, in Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS’04).

[33] Zitzler, E. and Thiele, L., “Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach”, IEEE Transactions on Evolutionary Computation, 3(4):257-271, 1999.