http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 6 (7): 978-986, 2007
ISSN 1812-5638

Genetic Load and Time Prediction Technique for Dynamic Load Balancing in Grid Computing

Zahida Akhtar
Department of Computer Science, Faculty of Basic and Applied Sciences,
International Islamic University Islamabad, Pakistan

Abstract: Gird computing is an emerging science in the field of distributed computing which involves coordinating and sharing computing, application, data, storage, or network resources across dynamic and geographically dispersed organizations. Scheduling and load balancing techniques are critical issues in grid computing for achieving good performance. The goal of load balancing is to minimize the response and execution time of a program by trying to equally spread the load on processors and maximizing their utilization. It has been proven that finding optimal schedules for the load-balancing problem is an NP-complete problem, even when the communication cost is ignorable. Genetic algorithms are a probabilistic search approach, which are founded on the ideas of evolutionary processes. They are particularly applicable to problems that are large, non-linear and possibly discrete in nature; features that traditionally add to the degree of complexity of solution. Present research aims to solve the grid load-balancing problem using Genetic Algorithms. A new Genetic Algorithm based task scheduling technique is introduced, which has been tested on a multi-node grid environment and the experimental results show that this new technique can lead to significant performance gain in various applications.

Key words: Genetic Algorithms (GA), load balancing, grid computing, distributed computing, heterogeneous computing, evolutionary computing

INTRODUCTION

Grid computing, as defined by its founders, has emerged as an important new field, distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications and, in some cases, high-performance orientation (Foster *et al*., 2001).

Grid computing is a new technology that transforms a computer infrastructure into an integrated, pervasive virtual environment for dynamic collaboration and shared resources anywhere in the world providing users, especially in science, with unprecedented computing power, services and information (Reddy, 2004). With the advance in technologies, the cost of computation resources required per operation is continuously decreasing. Grid computing is one of many factors which enable the effective use of wide spread computing resources thereby providing non-trivial services to users. According to the Department of Computer Science at the University of Warwick, with the emergence of grid environments featuring dynamic resources and varying user profiles, there is an increasing need to develop reliable tools that can effectively coordinate the requirements of an application with available computing resources. The ability to predict the behaviour of complex aggregated systems under dynamically changing workloads is particularly desirable, leading to effective resource usage and optimization of networked systems.

Scheduling and load balancing techniques are critical issues in grid computing for achieving optimum performance. The goal of load balancing is to minimize the response and execution time of a program by trying to equally spread the load on processors and maximizing their utilization. Present research aims to solve the grid load-balancing problem using Genetic Algorithms.

Genetic algorithms are based on a biological metaphor: They view learning as a competition among a population of evolving candidate problem solutions. A 'fitness' function evaluates each solution to decide whether it will contribute to the next generation of solutions. Then, through operations analogous to gene transfer in sexual reproduction, the algorithm creates a new population of candidate solutions (Melanie, 1999; Koza, 1990; Goldberg, 1989; Vose, 1999; Rennard, 2000). Genetic Algorithms are nondeterministic stochastic search/optimization methods that utilize the theories of evolution and natural selection to solve a problem within a complex solution space. They are computer-based problem solving systems which use computational models

of some of the known mechanisms in evolution as key elements in their design and implementation (Sandikci, 2000).

BASIC CONCEPT

Recent research studies indicate that genetic algorithms significantly enhance the performance of real time applications. In one such study done by Wu and Chau (2006), authors employ the hybrid genetic algorithm based artificial neural network model for flood prediction. The proposed model is tested against empirical linear regression model, conventional ANN model and a GA model and results reveals that proposed hybrid GA-based ANN algorithm outperforms the conventional models. The limiting factor of this research study is that it requires additional modelling parameters and longer computation time.

In another study, Chau and Albermani (2003) develop the prototype system by coupling the blackboard architecture, an expert system shell VISUAL RULE STUDIO and Genetic Algorithm (GA). Chau and Albermani (2003) proposed the said system for the optimize design of liquid retaining structures which can act as a consultant to assist novice designers in the design of liquid retaining structures. Near-optimal solutions are claimed to be achieved after exploration of small portion of search space at extraordinarily converging speed.

Another example of genetic algorithm in a real time application is, Usage of parallel genetic algorithm for multiple criteria rainfallBrunoff model calibration which is proposed by (Cheng *et al*., 2005). The method uses the fuzzy optimal model to evaluate multiple alternatives with multiple criteria where chromosomes are the alternatives, whilst the criteria are flood performance measures Cheng *et al*. (2005). The proposed approach produces the similar results when compared with results obtained by using a two-stage calibration procedure but it significantly reduces the overall optimization time and improves the solution quality. The disadvantage of this research study is that it splits the whole procedure into two parts which makes it difficult to integrally grasp the best behaviours of model during the calibration procedure. In Continuation to study Cheng *et al*. (2005 and 2006) proposed a new method to the multiple criteria parameter calibration problem, which combines GA with TOPSIS for Xinanjiang model. Cheng *et al*. (2006) removes the disadvantage of their previous research study and integrates the two parts of Xinanjiang rainfall-runoff model calibration together thus simplifying the procedures of model calibration and validation. Comparison of results with two-step procedure shows that the proposed methodology gives similar results to the previous method, is also feasible and robust, but simpler and easier to apply in practice.

In another study, Chau (2004) proposed a two-stage dynamic model to assist construction planners to formulate the optimal strategy for establishing potential intermediate transfer centers for site-level facilities such as batch plants, lay-down yards, receiving warehouses, various workshops, etc. Under the proposed approach, the solution of the problem is split into two stages, namely, a lower-level stage and an upper-level stage. Standard linear programming method is used to solve former stage whereas the latter is solved by a genetic algorithm. The efficiency of the proposed algorithm is demonstrated through case examples.

In this research study the problem area has been described, which is mostly based on the description of Dong and Akl (2006). While different approaches have used GAs for solving load balancing problems yet the issues that remain to be addressed can be broadly categorized as the following:

- The execution time for load balancing has not been considered or has not been quantitatively described.
- Most of the algorithms are restricted to static load balancing and as such require prior knowledge of various parameters. While this approach may work in problems of equivalent nature but cannot be broadly applied to different applications.
- A few dynamic load balancing algorithms that have been studied and are also mentioned in the literature review have not been implemented in loosely coupled systems such as grid computing.
- To the best of the author, s knowledge, no algorithm has been designed to prevent resubmission in case of load failure. The algorithms that incorporate fault tolerance use a simple strategy for restarting the task which in some cases requires extensive overheads.

Efficient execution in a distributed system can require, in the general case, mechanisms for the *discovery* of available resources, the *selection* of an application-appropriate subset of those resources and the *mapping* of data or tasks onto selected resources.

Grid computing has become an increasingly popular solution to optimize resource allocation in highly charged IT environments. In one of the recent research studies done by Wieczorek *et al*. (2005) three different algorithms (namely HEFT, Genetic and simple Myopic algorithm) are compared in terms of incremental versus full-graph scheduling for balanced versus unbalanced workflows.

Without considering effect of the typical network scenarios Wieczorek *et al.* (2005) declare HEFT as better algorithm. A multi-tiered framework based on, Globus providers, distribution brokers and local schedulers is used for grid work load management, out of which only lowest tier is primary focused by Spooner *et al.* (2003). Spooner and his colleague use iterative heuristic algorithm and performance prediction techniques for performance based upon global and local scheduling. The future work of Spooner *et al.* (2003) for examining the other two upper tiers is still on its way. Cao *et al.* (2005) addresses grid load balancing issues using a combination of intelligent agents and multi-agent approaches. The experimental result of research study of Cao *et al.* (2005) proves that the use of a distributed agent strategy can reduce the network overhead significantly and make the system scale well??? rather than using a centralized control, as well as achieving a reasonable good resource utilization and meeting application execution deadlines. Abraham *et al.* (2000) addressed the hybridization of the three popular nature, s heuristics namely Genetic Algorithms (GA), Simulated Annealing (SA) and Tabu Search (TS) for dynamic job scheduling on large-scale distributed systems but did not provide any experimental results for research evaluation. A novel mapping heuristic based on the Cross-Entropy (CE) method, for mapping a set of interacting tasks of a parallel application onto a heterogeneous computing platform, was proposed by Sanyal and Das (2005).

According to their research studies, Cross Entropy methods are inherently slow and this slowness of the CE based methods in generating the appropriate mapping can decrease the performance gain for a large set of tasks. Moreno (2003) proposed new rescheduling policies for job migration under cost constraints after analysing the main tasks that the grid resource broker has to tackle (like resource discovery and selection, job scheduling, job monitoring and migration etc.) in detail. Wagner and Affenzeller (2004) present a new environment for parallel heuristic optimization based upon the already proposed Heuristic-Lab in.

A Formal model which allows multiple schedule optimizations and a new efficient heuristic approach based on genetic algorithms and list scheduling is presented by Grajcar (2000). In spite of the fact that the algorithm contains some programming inefficiencies, it still performs well in terms of running speed and result quality. Zomaya and The (2001) investigate how a genetic algorithm can be employed to solve the dynamic load balancing problem. The dynamic load-balancing algorithm is developed by Zomaya and The (2001) whereby optimal or near-optimal task allocations can Aevolve@ during the operation of the parallel computing system. A scheduling routine based upon a genetic algorithm is developed (Greene, 2001) which is claimed to be very effective and has relatively low cost. Two important aspects of this research study are: loads on the processors are well balanced and scheduling per se remains cheap in comparison to the actual productive work of the processors. Dynamic Distributed Genetic Algorithm is proposed by Yi *et al.* (2000). According to the paper, dynamic distributed GA with directed migration has great potential to overcome premature convergence.

The contribution of Song *et al.* (2005) is two-fold: first the Min-Min and Sufferage heuristics are enhanced under three risk modes driven by security concerns and secondly a new Space-Time Genetic Algorithm for trusted job scheduling is proposed. The results of Song *et al.* (2005) shows that there is a need of more research study in order to over come the shortcoming of security driven Min-Min and Sufferage heuristics which are unstable when applied to different types of workloads. A novel GA-based approach is proposed by Kim and Weissman (2004) to address the problem of scheduling a divisible Data Grid application while considering communication and computation at the same time in wide area data intensive environment. According to Kim and Weissman (2004) the results from the experiments on GA-related parameters suggest that the initialization of population with chromosomes of good quality is critical to GA-based approach in terms of the quality of solution and the convergence rate. But the problem of multiple jobs competing for shared resources has not been overcome in this study. Five heuristics that have been designed, developed and simulated using the HC environment, are presented by Shivle *et al.* (2005). Application tasks are composed of communicating subtasks with data dependencies and multiple versions were mapped using the heuristics described by Shivle *et al.* (2005) and the results can be used in the development of ad hoc grids. The EVOLVE/G system, which is a Grid tool for developer of evolutionary computation, is proposed by Tanimura *et al.* (2002). This system consists of an Agent and multiple workers. Since the data can be exchanged between the Agent and Workers freely, any logical models of EC can be integrated. Jing *et al.* (2004) describes a parallel hybrid-GA (PHGA) for combinatorial optimization using an island model running in a networked computing environment. Jing *et al.* (2004) opens several issues for future research like extensive study on scalability of parallel GA in a distributed computing framework. Applicability of parallelizing the local search of a serial GA and other heuristics for local search can be

explored to enhance the performance of the parallel GA. Cao *et al.* (2003) developed a GA-based scheduler for fine-grained load balancing at the local level, which was then coupled with an agent-based mechanism that was applied to load balance at a higher level. Future enhancement to the system will include the integration with other grid toolkits (e.g., Globus MDS and NWS).

MATERIALS AND METHODS

In this research work an attempt has been made to increase the efficiency of grid scheduler. GAs based scheduling algorithm named Dynamic Online Scheduling is proposed for better resource optimization and task scheduling. The scheduling process in this algorithm is addressed in two layers namely pre- scheduling and post-scheduling. The newly coming problems from outside grid boundary are scheduled in the first layer which is pre-scheduling. In post- scheduling the load balancing of the already submitted tasks is done, that is if a certain resource is found overloaded with work while some other resources are free then some of the jobs of the overloaded machine are automatically shifted to the free machines while keeping in mind the robustness, reliability and efficiency of the job as well as the time cost, communication cost. Resource cost will also be considered.

Load and time prediction technique: The load on the resources and the execution time of the tasks both are interrelated and depend on each other. Every task has certain execution time and every task puts a certain amount of load on the machine it is executed on. We have developed a mechanism which on the basis of task attributes (e.g., size, type etc.) and resource attributes (e.g., memory, cpu cycles, load) tells the task execution time on that machine. This strategy gives us the advantage of dynamicity in the pool of heterogeneous resources as well as tasks, since both task and resource attributes are not assumed to be fixed. The load and time prediction strategy is based on historical data. Each resource is continuously monitored for its utilization and the data is entered into the log file. This log file helps us to set the threshold for our resources at any particular hour of day. The resource threshold is updated after each hour. At the same time an extensive amount tasks are executed on each resource and the resource utilization is calculated with each task and the log file logs that task = s attributes and resource utilization. After that, we rearrange the task log file according to the task attributes and time of the day they were submitted on. Then we give this data to the genetic algorithms to learn which resource is best suited for which kind of task at any particular given time of the day.

So far time prediction seems enough for load balancing since each task will be assigned on the basis of threshold plus the current situation of resource. But that, s not exactly what happens in the real world. When multiple tasks come the scheduler calculates the task execution time for each task with respect to the current situation of the available resources it is assigned to. Each task will definitely increase some amount of load on a resource. If multiple tasks are assigned to a resource then after one task starts running, the current situation of the resource will change but the scheduler predicts the task execution time according to the pervious resource situation that is before the execution of the previous task began. To cover up this flaw, we developed a Load Prediction strategy. This strategy is also based on history data but this data is required to be generated only once for each different set of attributes of task.

At this point it all seems static at compile time work while its not. Dynamicity comes when a new task arrives, GA scheduler become is activated and it retrieves the required task attributes, lists the currently available resources and predicts the execution time of the task for each resource on the basis of current resource parameters (e.g., memory, cpu cycles, load) or predicted parameters, task attributes and the history data from log file (that how much time the task of given attributes takes to execute with respect to the resource history parameters). The formula used by GAs to calculate the prediction time is explained in a later section.

Dynamic online scheduling: In dynamic online scheduling, scheduling is done on the basis of the current situation of the grid, which takes the current resources states/parameters from the resource collector and task list to be scheduled from the task collector and provides both lists to GA based scheduler as shown in Fig. 1. The Dynamic Online Scheduling procedure works in the following manner.

Pre-scheduling

- A list of available resources is generated on the basis of the current situation of resources as well as the history data about resources.
- A list of tasks to be scheduled is generated from the task queue.
- Both lists (Resource list and Task list) are provided to the GA based Analyzer and Load Balancer which will generate the optimized mapping of tasks to resources.
- Mapping Engine assigns the tasks to resources.
- Task Executer executes the task and displays the output.

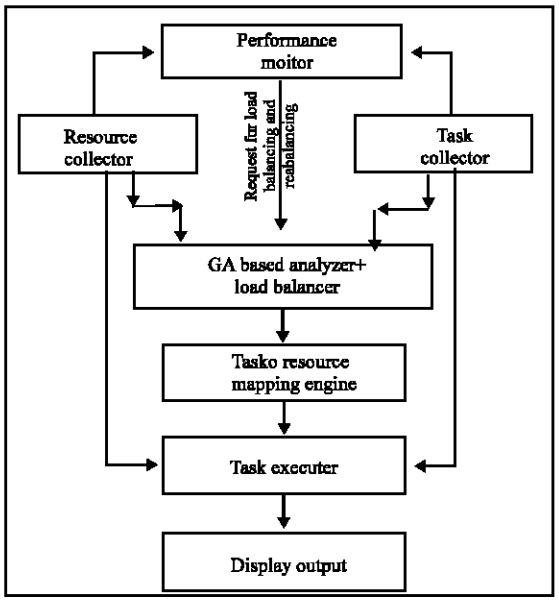

Fig. 1: Dynamic online scheduling overview

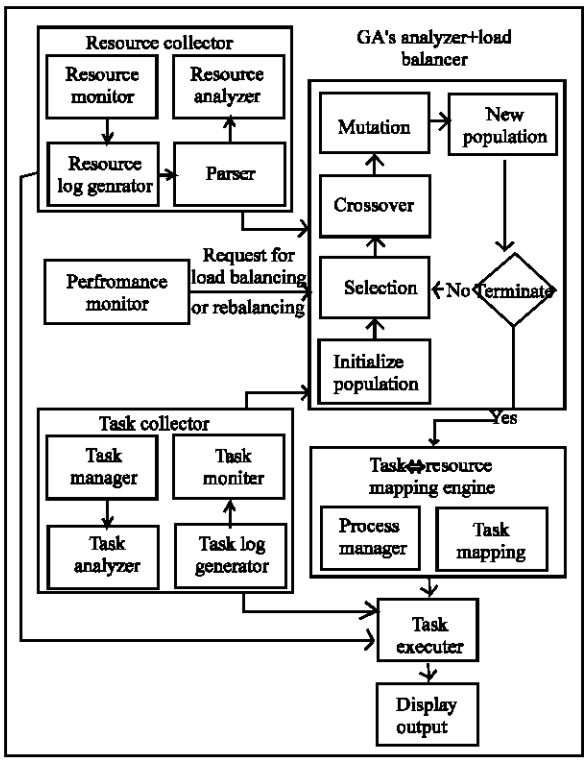

Fig. 2: Detailed internal architecture of online scheduling

Post-scheduling

- Post scheduling algorithm becomes active when the performance monitor views that certain resources are being over utilized while some others are being under utilized. The criteria for the measurement of the resource, s = overload (that is if whether a certain resource is overloaded or not) is its threshold limit.

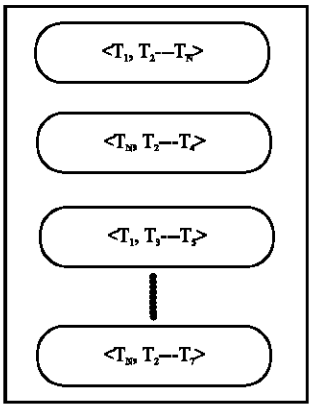

Fig. 3: Chromosome representation

- Each resource has been assigned a certain threshold on the basis of history data.
- If certain resources are overloaded then the performance monitor comes into action. It will place the request to Load Balancer for redistribution of the tasks.
- Load Balancer will activate the Resource Collector to provide the list of overloaded resources and under utilized resources and Task Collector to provide the Task list of overloaded resources.
- Load Balancer will then generate the new mapping.

Block diagram for pre and post scheduling has provided the detailed internal architecture of system as shown in Fig. 2.

Genetic algorithms for scheduling

Chromosome representation: For applying GAs directly or coupled with other meta-heuristics, problem (chromosome) representation is very important and it directly affects the performance of the proposed algorithm. The first decision a designer has to make is how to represent a solution in a chromosome. We assume that the Tasks and Resources are arranged in an ascending order according to the Task attributes (that is size, type, submission time) and Resource usage (that is least loaded comes first). Figure 3 depicts the chromosome representation which is used in our current strategy. Basically each chromosome in the population contains permutation of tasks and their fitness is calculated according to the resources they were assigned to. Task T_1 is allocated to resource R1, T_2 to R2 and T_3 to

R3 and so on. When T_1 is completed, resource R1 is empty and task T_N is allocated. This procedure goes on until all the tasks are allocated.

GA approach for task scheduling and load balancing

- Get the Task List from the Task Collector of length TNT where TNT is the total number of tasks to be scheduled.
- Get the Resource List from the Resource Collector of length TNAR where TNAR is the total number of available resources, if no resource is available then wait until resources become available.
- At t = 0; generate an initial population with P chromosomes Popi(t), where P represents the permutations of tasks and is calculated as:

$$P = {}_{TNT}P_{TNT}$$

- For each chromosome (I = 1 to P), first allocate the jobs to the available resources based on the FCFS basis. Then calculate the predicted time for each task according to the current parameters of the resource it is assigned to from the task history log. For example if one parameter of a resource say load is used then the task time prediction formula will be as follows:

$$\frac{PT_{T,R}}{PL_R} = \frac{HT_{T,R}}{HL_R}$$

$$PT_{T,R} = PL_R \frac{HT_{T,R}}{HL_R}$$

Where:

$PT_{T,R}$ = Predicted completion time of task T on resource R

PL_R = Present load on resource R for 1st task and Predicted Load for next assigning tasks (Each task will increase some load on the resource which will be calculated as predicted present load for next task)

$HT_{T,R}$ = History completion time for task T on resource R (taken from Task Log History).

HL_R = History load on resource R when task T had completion time $HT_{T,R}$

Now the predicted time for each task in the chromosome can be easily calculated using afore mentioned formula.

- Fitness value for each chromosome is calculated which will tell us the make-span of the schedule. Fitness value is calculated using following formula:

$$F = \frac{1}{Max(\sum_{i=1}^{TNT} PT_{Ti,R})}$$

- The make-span of the schedule is calculated using following formula:

$$\emptyset = Max(\sum_{i=1}^{TNT} PT_{Ti,R})$$

Where

$$\sum_{i=1}^{TNT} PT_{Ti,R}$$

represents the total number of tasks assigned to resource R.

- Apply Crossover operator on the population according to the probability selected, Crospop(t+1) = recombined chromosomes of the population $NewPop_i$ (t+1);
- Apply mutation operator on the population according to the probability selected, MutPop (t+1) = mutated population CrosPop (t+1).
- Evaluate the fitness of each chromosome in the new population and check if the specified fitness value is achieved or not. If not, start the GAs loop again.
- Send the specified schedule generated by GAs to the Mapping Engine which will assign the tasks to the resources. In case of post scheduling the Mapping Engine will migrate the tasks from the overloaded resources to relatively less loaded resources.

RESULTS AND DISCUSSION

As a fundamental base, we have adopted OpenMosix as the resource monitoring and management tool. OpenMosix provides the update network weather service as well as logging the utilization information of each resource. A Task Management tool is developed which receives task from the users. GA scheduler works at the back end which takes tasks form task management tool and maps them to the available resources (List of available resources is provided by resource management tool i.e., OpenMosix) First we run bunches of tasks on limited resources (2-8 machines) without any involvement of GA scheduler and calculate the execution cost that is the total time it takes to complete all the assigned tasks on all machines and then we perform the same test using GA based scheduler and again calculate the execution cost. We calculate the execution cost for different sizes of schedules by slowly increasing the number of resources (5, 10, 15, 20, 25 and 30 number of machines) both with and without GA and the results shows that performance of system increases when GA base scheduler is used. Figure 4 shows the comparison results of system with proposed GA scheduler and without GA scheduler. Schedule Size vs. Make span is plotted in Fig. 4.

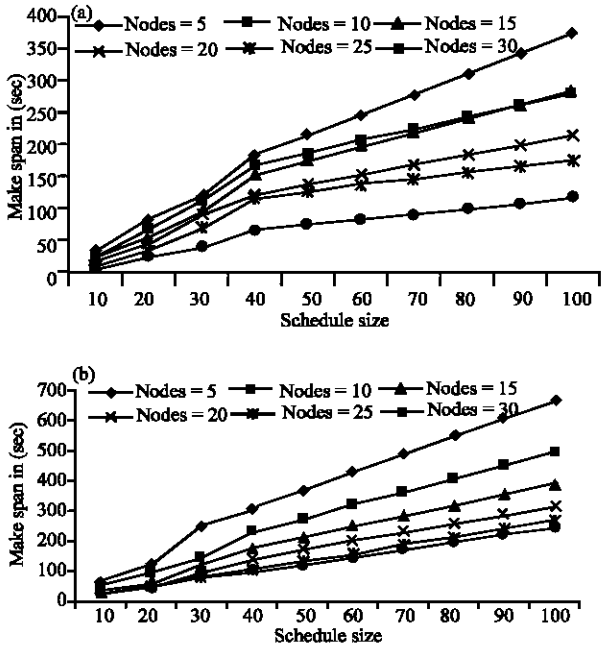

Fig. 4: (a) System performance with GAs based scheduler schedule size vs. time (b) System performance without GAs based scheduler schedule size vs. time openmosix dynamic scheduler

Table 1: Make span time in seconds generated by GA on different sizes of schedule and number of processors

	No. of processors					
S. size	5	10	15	20	25	30
10	31.8	21.8	18.9	15.6	10.2	8.5
20	81.1	66.7	53.4	44.7	33.8	23.3
30	119.7	111.0	96.3	90.9	69.9	40.2
40	183.4	167.4	154.8	122.2	115.5	64.9
50	215.2	189.3	173.7	137.9	125.7	73.5
60	247.1	211.1	192.6	153.6	136.0	82.0
70	279.0	233.0	211.5	169.2	146.2	90.6
80	310.9	254.9	230.5	184.9	156.4	99.1
90	342.8	276.8	249.4	200.6	166.6	107.6
100	374.7	298.7	268.3	216.3	176.8	116.2

Table 2: Make span time in seconds generated by openmosix on different sets of schedule size and number of processors

	No. of processors					
S. size	5	10	15	20	25	30
10	59.0	043.8	036.5	030.0	27.5	25.5
20	121.0	094.5	057.6	050.0	48.0	45.0
30	251.5	141.0	119.5	092.4	82.5	77.0
40	310.5	232.5	174.5	139.0	102.5	93.6
50	369.5	276.3	211.0	169.0	130.0	119.1
60	428.5	320.1	247.5	199.0	157.5	144.6
70	487.5	363.9	284.0	229.0	185.0	170.1
80	546.5	407.7	320.5	259.0	212.5	195.6
90	605.5	451.5	357.0	289.0	240.0	221.1
100	664.5	495.3	393.5	319.0	267.5	246.6

Table 3: Performance gain of Genetic algorithms based scheduler over openmosix scheduler

	No. of processors				
S. size	10 (%)	15 (%)	20 (%)	25 (%)	30 (%)
10	49.96	51.83	52.27	37.13	33.47
20	70.67	92.75	89.50	70.59	51.92
30	78.75	80.63	98.44	84.83	26.27
40	72.01	88.74	87.95	112.76	69.44
50	68.52	82.35	81.62	96.76	61.74
60	65.98	77.85	77.19	86.35	56.75
70	64.05	74.51	73.93	79.04	53.26
80	62.53	71.92	71.42	73.61	50.68
90	61.31	69.87	69.43	69.43	48.70
100	60.31	68.20	67.82	66.11	47.12

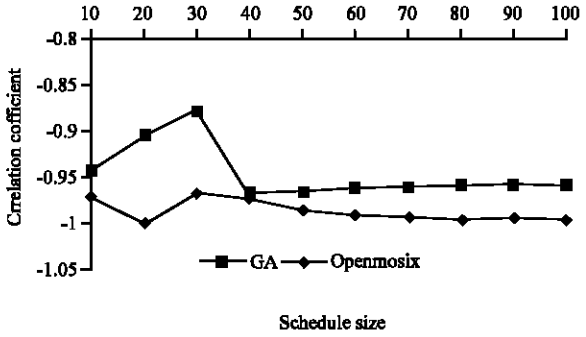

Fig. 5: Correlation coefficient of GA scheduler and Openmosix scheduler

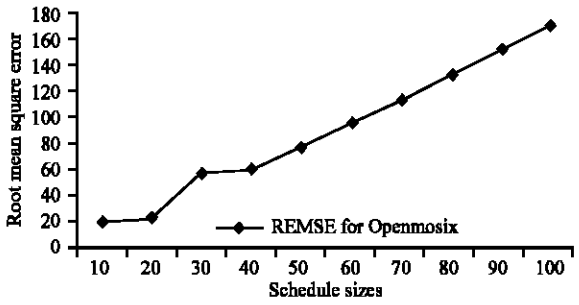

Fig. 6: Root mean square error for Openmosix scheduler with respect to GA based for different schedule size

Table 1 and 2 displays the detailed information of execution time of different sizes of schedules with respect to different number of resources generated by GA scheduler and Openmosix scheduler, respectively. Table 3 indicates the performance gain of GA scheduler over Openmosix scheduler for each set of schedules and resources.

In order to visualize the performance of system more clearly root mean square root error and correlation coefficient is calculated from the results generated by

Table 4: Improvement in the Make span (Schedule Cost) after each 100 generations for different schedule sizes

No. of generations	Schedule Size 70	80	90	100
100	427.1055	437.8263	484.9545	491.3856
200	323.7534	356.622	327.1157	374.9545
300	219.8443	234.5184	225.4005	280.4443
400	174.4709	128.2745	124.9915	191.7306
500	90.6014	99.1374	107.6734	116.2094
600	90.6014	98.1301	106.5564	115.5494
700	89.0151	98.1301	106.5564	115.5494
800	89.0151	97.1404	106.5564	115.5494
900	88.0151	97.1404	106.0112	115.0013
1000	88.0151	97.1404	105.5461	115.0013
1100	87.6014	96.0011	105.5461	114.5095
1200	87.6014	96.0011	105.5461	114.5095
1300	86.0121	96.0011	105.5461	114.5095
1400	86.0121	96.0011	105.0215	114.1421
1500	86.0121	96.0011	105.0215	114.1421
1600	85.0501	96.0011	105.0215	114.1421
1700	85.0501	96.0011	105.0215	114.1421
1800	85.0501	96.0011	105.0215	114.1421
1900	85.0501	96.0011	105.0215	114.1421
2000	85.0501	96.0011	105.0215	114.1421

Openmosix scheduler and GA scheduler. Figure 5 shows the correlation coefficient series, for different sizes of schedules and number of resources, of both GA scheduler and Openmosix Scheduler and it can be clearly seen that GA scheduler is more correlated when we increase the schedule size over time than Openmosix scheduler.

Figure 6 displays the root mean square root error of Openmosix scheduler with respect to GA scheduler and the plot indicates that as the schedule size increases the error generated by Openmosix also increases.

Finer results are obtained when GAs is executed for another 1000-1500 generations but by increasing the number of generations performance gain due to GAs starts compromising in terms of scheduling cost so in order to obtain the best optimal result number of generations for GAs are set to fixed for 500. After running for 1500 generations the results appear more or less the same. Table 4 shows the improvement in Make span ? (schedule cost) after each 100 generations for different sizes of schedules.

CONCLUSIONS AND FUTURE WORK

In this research study we attempt to address the problem of load balancing in grid computing using one of the popular heuristics namely GAs. Genetic Algorithm predicts the execution time for each task with respect to the resource it is assigned to. The prediction time is based on the current attributes of task, current and historical parameters (like load, memory etc) of resources. We have tested this algorithm on a 30 machines heterogeneous grid environment with different schedule sizes and the results show that the proposed strategy can lead to the significant performance gain.

Future work will examine the application of proposed GA based algorithm as parallel genetic algorithms and dynamic distributed algorithms proposed by Yi *et al.* (2000). In both cases number and size of populations must be carefully determined. Even though significant progress has been made in modelling the infrastructure of grid computing but a close review clearly indicates that not much progress is made in formulating the efficient and globally optimized, grid-scheduling algorithm for allocating jobs (Abraham *et al.* 2000). Therefore we are planning to investigate this research area in depth using GAs as well as other heuristic algorithms.

REFERENCES

Abraham, A., R. Buyya and B. Nath, 2000. Nature's heuristics for scheduling jobs on computational grids. The 8th IEEE International Conference on Advanced December 14-16, 2000. Cochin, India.

Cao, J., D.P. Spooner, S.A. Jarvis, S. Saini and G.R. Nudd, 2003. Agent-based grid load balancing using performance-driven task scheduling. Proceedings of the 17th International Symposium on Parallel and Distributed Processing, IEEE Computer Society, 2003. Washington DC, USA.

Cao, J., D.P. Spooner, S.A. Jarvis and G.R. Nudd, 2005. Grid load balancing using intelligent agents. Future generation computer systems, Vol. 21. Elsevier Science Publishers B.V. Amsterdam, The Netherlands.

Cheng, C.T., X.Y. Wu and K.W. Chau, 2005. Multiple criteria rainfall-runoff model calibration using a parallel genetic algorithm in a cluster of computer. Hydrol. Sci. J., 50: 1069-1087.

Chau, K.W. and F. Albermani, 2003. Knowledge-based system on optimum design of liquid retaining structures with genetic algorithms. J. Struc. Eng. ASCE., 129: 1312-1321.

Chau, K.W., 2004. A two-stage dynamic model on allocation of construction facilities with genetic algorithm. Automation in Construction, 13: 481-490.

Cheng, C.T., M.Y. Zhao, K.W. Chau and X.Y. Wu, 2006. Using genetic algorithm and TOPSIS for Xinanjiang model calibration with a single procedure. J. Hydrol., 316: 129-140.

Dong, F. and S.G. Akl, 2006. Scheduling Algorithms for Grid Computing: State of the Art and Open Problems. Technical Report No. 2006-504, School of Computing, Queen, s University, Kingston, Ontario. http://www.dcs.warwick.ac.uk http: //cs. felk.cvut. cz /~xobitko /ga/ about.html

Foster, I., C. Kesselman and S. Tuecke, 2001. The Anatomy of the Grid Enabling Scalable Virtual Organizations. Int. J. High Performance Comput. Appl. Vol 15, Sage Publications, Inc. Thousand Oaks, CA, USA., pp: 1.

Goldberg, D.E., 1989. Genetic Algorithms in search optimization and Machine Learning. Pearson Education Pte. Ltd., Singapore.

Grajcar, M., 2000. Conditional scheduling for embedded systems using genetic list scheduling. Proceedings of the 13th International Symposium on System Synthesis, 2000. Washington DC, USA. IEEE Computer Society.

Greene, W.A., 2001. Dynamic load balancing via a genetic algorithm. Proceedings of the 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01), 2001. Washington DC, USA. IEEE Computer Society.

Jing, T., M.H. Lim and Y.S. Ong, 2004. Island model parallel hybrid-ga for large scale combinatorial optimization. The Eighth International Conference on Control, Automation, Robotics and Vision (ICARCV2004), Special Session on Computational Intelligence on the Grid, December 6-9, 2004. Kunming, China.

Kim, S. and J.B. Weissman, 2004. A GA-based approach for scheduling decomposable data grid applications. Proceedings of the International Conference on Parallel Processing (ICPP'04) B Vol. 00, 2004. Washington DC, USA. IEEE Computer Society.

Koza, J.R., 1990. Genetic programming: A paradigm for genetically Breeding populations of computer programs to Solve problems. Stanford University Computer Science Department technical report STAN-CS-90-1314.

Melanie, M., 1999. An Introduction to Genetic Algorithms. A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England.

Moreno, R.A., 2003. Job scheduling and resource management techniques in dynamic grid environments. 1st European Across Grids Conference, 2003. Heidelberg, Germany.

Reddy, S., 2004. Grid computing: Crossing the Chasm, 24th September 2004. http:// sciencecareers. sciencemag.org/carrer_development/previous_issues/articles/grid_computing_crossing_the_chasm/(parent) /12093.

Rennard, J.P., 2000. Genetic Algorithm Viewer: Demonstration of a Genetic Algorithm. http://www.rennard.org/alife.

Sandikci, B., 2000. Genetic Algorithms. http:/ /www.ie. bilkent. edu.tr/~lors/ie572/burhaneddin.pdf .

Sanyal, S., and S.K. Das, 2005. MaTCH: Mapping data-parallel tasks on a heterogeneous computing platforms using the cross-entropy heuristic. Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, 2005. Washington DC, USA. IEEE Computer Society.

Shivle, S., H.J. Siegel, A.A. Maciejewski, T. Banka, K. Chindam *et al*., 2005. Mapping of subtasks with multiple versions in a heterogeneous ad hoc grid environment, parallel computing, Vol. 31, Elsevier Science Publishers B.V. Amsterdam, The Netherlands.

Song, S., Y.K. Kwok and K. Hwang, 2005. Security-driven heuristics and A FAST genetic algorithm for trusted grid job scheduling. Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, 2005. Washington DC, USA. IEEE Computer Society.

Spooner, D.P., S.A. Jarvis, J.Cao, S. Saini and G.R. Nudd, 2003. Local grid scheduling techniques using performance prediction. IEEE Proceedings Computers and Digital. Techniques, 150. April 2003, Institution of Electrical Engineers, Great Britain.

Tanimura, Y., T. Hiroyasu, M. Miki and K. Aoi, 2002. The System for Evolutionary Computing on the Computational Grid. Proceedings of Parallel and Distributed Computing and Systems, 2002.

Vose, M.D., 1999. The Simple Genetic Algorithm, foundation and theory. A Bradford Book, The MIT Press. Cambridge, Massachusetts. London, England.

Wagner, S. and M. Affenzeller, 2004. Heuristic lab grid B a flexible and extensible environment for parallel heuristic optimization. Proceedings of the 15th International Conference on Systems Science, Vol. 1, 2004. Oficyna Wydawnicza Politechniki Wroclawskiej.

Wieczorek, M., R. Prodan and T. Fahringer, 2005. Scheduling of Scientific Workflows in the ASKALON Grid Environment. ACM SIGMOD Record, Vol. 34. ACM Press. New York, USA.

Wu, C.L. and K.W. Chau, 2006. A flood forecasting neural network model with genetic algorithm. Int. J. Environ. Pollut., 28: 261-273.

Yi, W., Q. Liu and Y. He, 2000. Dynamic distributed genetic algorithms. Proceedings of the 2000 congress on evolutionary computation CEC00, 2000. California, USA. IEEE Press.

Zomaya, A.Y. and Y.H. Tan, 2001. The observations on using genetic algorithms for dynamic load balancing. IEEE Trans. Parallel Distributed Syst., Vol. 12 IEEE Press.