Genetic algorithm in Grid Scheduling with multiple objectives

SIRILUCK LORPUNMANEE[1], MOHD NOOR MD SAP[2],
ABDUL HANAN ABDULLAH[3], SURAT SRINOY[4]
[1,4]Faculty of Science and Technology, Suan Dusit Rajabhat University
295 Rajasrima Rd., Dusit, Bangkok, Thailand,
Tel: (662)-2445244, Fax: (662)-6687136
http://www.dusit.ac.th

[1,2,3]Faculty of Computer Science and Information Systems, University Technology of Malaysia, 81310 Skudai, Johor, Malaysia,
Tel: (607) - 5532070, Fax: (607) 5565044
http://www.utm.my

Abstract: - Grid computing is the principle in utilizing and sharing large-scale resources to solve the complex scientific problem. Under this principle, Grid environment has problems in providing flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources. However, the major problem is in optimal job scheduling, which Grid nodes need to allocate the resources for each job. This paper proposes the models for multi-objective jobs scheduling in Grid environment. The model presents the strategies of allocating jobs to different nodes. We develop the models based on multi-objective of genetic algorithm to select multiple optimization scheduling of the jobs. In this preliminary tests we show how the solution founded may minimize the average waiting time and the make-span time in Grid environment. The benefits of the usage of multiple objective genetic algorithm is improving the performance of the scheduling is discussed. The simulation has been obtained using historical information to study the job scheduling in Grid environment. The experimental results have shown that the scheduling system using the multiple objective genetic algorithms can allocate jobs efficiently and effectively.

Key-Words: - Grid computing, Genetic algorithm, job scheduling, optimal job scheduling, average waiting time, make-span time.

1 Introduction

Grid Computing is the principle that occurs for a long period of time by focusing on virtual organizations [1] to share large-scale resources, innovating applications and in some cases getting high-performance orientation. Under this principle, Grid has problem in flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources. In Grid [2] concept is a new generation of technologies combine physical resources and applications that provide vastly more effective solutions to complex problems (e.g., scientific, engineering and business). These new technologies must be built on secure discovery, jobs allocates to resources, integration resources and services from the others. In [3] is a formal definition of Grid concepts. They define conceptual models is abstract machines that support applications and services. Fig. 1 taken from [3], are formally defined (e.g., Organization, Virtual Organization, Virtual Machine, Programming System, etc.). Currently, Global Grid Forum [4] formulated and provided standards documents of virtual organization.

Grid Computing goes beyond distributing and sharing resources to more applications, which are adapting to use Grid resources. Although, using distributed resources are useful but this is possible only if the Grid resources are scheduled as well. The optimal scheduler will result in high performance in Grid computing but the poor scheduler will be making contrast result. Now, the Grid scheduling is big topic in Grid environment for new algorithm model**.**

The Grid scheduling is responsible for resource discovery, resources selection, and job assignment over a decentralized heterogeneous system, which resources belong to multiple administrative domains. Normally, the resources are requested by a Grid application, which use to computing, data and network resources etc. However, Grid scheduling of applications is absolutely more complex than scheduling an applications of a single computer. Because resources information of single computer scheduling is easy to get information, such as CPU frequency, number of CPU a machine, memory size, memory configuration and network bandwidth etc. But Grid environment is dynamic resources sharing and distributing. Then an application is hard to get resources information, such as CPU load, available memory, available network capacity etc. And Grid environment also hard to classify jobs characteristic, that run in Grid. There are basically two approaches to solve this problems, the first is based on jobs characteristic and second is based on a distributed resources discovery and allocation system. It should optimize the allocation of a job allowing the execution on the optimization of resources. The scheduling in Grid environment has to satisfy a number of constraints on different problems.

We have defined a set of them to study the feasibility and the usefulness of applying genetic algorithm to this field. The model are designed for multiple objective in scheduling system in which often involve a lot of historical data and many complex objective. Our model consider in job submit time, run time, idle time and jobs end time that each jobs running on the Grid. Genetic algorithms are applied to solve the jobs scheduling system with multiple objective and the results have shown that the scheduling system using the multiple objective genetic algorithm can improve the performance and can allocate jobs efficiently and effectively.

In this paper we discussed multiple objective genetic algorithm in scheduling system. To this end, we are reviewing a related work in Section 2. Next, Section 3 we brief the classification scheduling. Section 4, show the internal resources selection. Section 5, we show genetic algorithms and our model. Section 6, show the configuration and the result of experiment. Section 7 concludes the paper by summarizing this work, and providing a preview of future research in this area, is given in Section 8

2 Related worked

Most of the job scheduling in Grid environment based on job execute time and job run time has been proposed. In [5], the module prediction engine is a part of scheduling and offer a history based approach for estimating the run time of job submission. Intelligence will be a key feature in the next generation of Grid environment. In [6] proposed two modules for predicting the completion time of jobs in a service Grid and applying genetic algorithm to job scheduling. The problems of scheduling system on the Grid environment has been in [7], [8], [9], [10]. All of them adopt the method of genetic algorithm for jobs scheduling by applying different jobs characterization to improve performance. We noticed that their methods focused on optimization or sub-optimization for scheduling system.

Efficacious and effective job scheduling in Grid requires to model can allocate the available resources on Grid nodes to compute jobs, determine the current workload and predict the job execution time. In [11], was job scheduling in parallel and cluster computing; their goals are to achieve best performance and load balancing across the entire system.

Facing varying situations, intelligent Grid environments need complicated scheduling strategies and algorithms to handle different kinds of jobs.

Heuristic algorithms are often used in Grid environment for scheduling system. The algorithms use historical data of workload and explicit constraints to scheduling jobs [12], [13].

In [14] proposed models for scheduling system by using genetic algorithm with multiple goals. It’s considered problems in parallel machine scheduling.

Our approach, especially examine the implications of the fact that workload of jobs is expected to have an impact on the resources utilization and, even more interestingly for researcher on the performance quality. We use information about static workload data from the Standard Workload Archive [14] and it has been experimented in several publications [15], [16]. These workload traces consists of information about all job submissions on a machine for a certain period of time which usually ranges over several months and several thousands of jobs. Therefore, it is reasonable to start with the available workload traces information from the compute centers to evaluate the impact of jobs characterization in Grid environment.

3 Classification

The structure of the hierarchical classification of the taxonomy is shown in Fig. 2, taken from [17]. In the following subsections, we will briefly discuss the various design options for the allocation strategy.

3.1 Local vs global scheduling

At the highest level, general scheduling policies can be divided as local and global scheduling. Local scheduling algorithms are usually referred to the assignment of jobs to the time-slices of a single processor. As Global scheduling algorithms are normally referred to the assignment of jobs to the processors in parallel system.

3.2 Static vs dynamic allocation

Processors allocation can be dynamic or static depending on the time at which the allocation is done. In static scheduling algorithms, all of information regarding the jobs must be entirely known before execution time. Therefore, the processor that will execute the process is decided right at this time. As dynamic scheduling algorithms, all of information regarding the jobs is usually unknown before the jobs are executed in the system. Therefore, the scheduling decisions are made on the fly.

3.3 Distributes vs centralized allocation

Distributed scheduling algorithms, interact with each other and commit jobs to remote system. No central scheduling control is responsible for the scheduling system, as all of information on the state of available system in centralized scheduling algorithms must be collected here.

3.4 Adaptive vs non-adaptive allocation

An adaptive scheduling algorithms change its scheduling decisions in response to the previous and current behavior of the system. Normally, adaptive scheduling algorithms are dynamic. As non-adaptive scheduling algorithms does not change its scheduling decisions according to the previous and current behavior of the system.

4 Internal resources selection

Large parallel computing and heterogeneous computing system are usually shared by many users. The mechanisms that support this sharing follow as.

4.1 Space-sharing

The machine may be partitioned into sets of processors and each processor is allocated to a single job that is allowed to run to completion.

4.2 Time-sharing

More than one job may be allocated to a processor, in which case each job runs on some quantum time before being preempted to allow other jobs to run.

In our simulation we assumed that each of jobs is allowed to run in each node by using space-sharing mechanism. In space sharing scheduling, a job requests a fixed number of nodes on arrival and the scheduling finds and allocates the nodes, possibly after the job has been in a waiting queue. The job executes on the available nodes until completion. As the nodes are not shared and the jobs are never preempt during execution. To simulate, we only need to know the submission times, the run times, the idle times and the end times of jobs. In most modern parallel machines, the physical location of the nodes on which job executes does not significantly affect the execution, and then the workload traces can be taken as the execution time for simulations. All scheduling optimizations evaluated in this paper are for a space sharing model.

5 Genetic Algorithms and our Model

Genetic algorithms (GA) are a class of stochastic search algorithms which based on biological evolution [18], [19]. A basic GA can be represented as in Fig 7.2 taken from [20].

Genetic algorithms combine the exploitation of past results with the exploration of new areas of the search space by using survival of the fittest techniques combined with a structured of randomized information exchange, a GA can mimic some of the innovative intelligence of human search. A generation is a collection of artificial creatures (strings). In every new generation a set of strings is created using information from the previous times. Occasionally a new part is tried for good measure. GAs are randomized, but they are not simple random walks. They efficiently exploit historical information to speculate on new search points with expected improvement.

The approach used in this work generates a set of initial scheduling, evaluates the scheduling to obtain a measure of fitness, selects the most appropriate and combines then together using operators (crossover and mutation) to formulate a new set of solutions.

The basic type of GAs, known as the simple GA (SGA), uses a population of binary strings, single point crossover, and proportional selection

[18], [19]. Many other modifications to the SGA have been proposed; some of these are adopted in our work by using multi-objective. The following subsections explain the steps in our proposed approach;

5.1 Population

Typically, SGA use of a population of 30-100 individual solutions, as in our simulation use a very large population of 500 – 1,000 individuals in order to simulate.

5.2 Initialization and Realization

The first step in the GA is to create an initial population. Usually a random number generator is used to uniformly distribute numbers in the desired range. For our simulation, we use workload traces consist of jobs population and then chromosomes are

$$\begin{array}{l}
\{J_{s_1,r_1,i_1},\ J_{s_1,r_1,i_2},\ J_{s_1,r_1,i_3},\ \ldots,\ J_{s_1,r_1,i_I},\\
J_{s_1,r_2,i_1},J_{s_1,r_2,i_2},\ J_{s_1,r_2,i_3},\ \ldots,\ J_{s_1,r_2,i_I},\\
\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots\\
J_{s_1,r_R,i_1},J_{s_1,r_R,i_2},\ J_{s_1,r_R,i_3},\ \ldots,J_{s_1,r_R,i_I}\\
J_{s_2,r_1,i_1},\ J_{s_2,r_1,i_2},\ J_{s_2,r_1,i_3}\ ,\ \ldots,\ J_{s_2,r_1,i_I}\\
J_{s_2,r_2,i_1},J_{s_2,r_2,i_2}\ ,\ J_{s_2,r_2,i_3}\ ,\ \ldots,\ J_{s_2,r_2,i_I}\\
\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots\\
J_{s_2,r_R,i_1},J_{s_2,r_R,i_2}\ ,\ J_{s_2,r_R,i_3}\ ,\ \ldots,\ J_{s_2,r_R,i_I}\\
\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots\\
J_{s_S,r_R,i_1},J_{s_S,r_R,i_2}\ ,\ J_{s_S,r_R,i_3}\ ,\ \ldots,\ J_{s_S,r_R,i_I}\ \}.
\end{array}$$

Figure 1 Chromosome of job in Grid

Let *s* be the job submission time, *r* be the job run time and *i* be the job idle time.

A variation to this is the extended random initialization, where the GA is seeded with individuals known to be in the vicinity of the global minimum.

5.3 The Fitness and Objective Functions

The objective function provides the mechanism for evaluating each chromosome in the problem domain. In this case of a minimization problem, the most fit individuals would have the lowest numerical value for their objective function. The fitness function normalizes the objective function value. In our model is multi-objective genetic algorithm, then our fitness function are:

$$T_{idleTime} = \sum_{j=2}^{p} |ST_j - ET_i| \cong 0 \quad ;j > i \qquad (1)$$

Let ST_j be the submission time of the j-th job and let ET_i be the end time of the i-th job. Equation (1) represents the idle time of all the jobs where the aim of the process is to minimize the job idle time with respect to the scheduling.

Let *WT* be the waiting time of the j-th and i-th job and let *AWT* be the average waiting time of jobs. We define

$$AWT_{Currentchr\ omosome} \ge |WT_j - WT_i| \ ;j > i \qquad (2)$$

Equation (2) represents the average waiting time and waiting time of each job in Grid environment where the aim of the process is to minimize the average waiting time and waiting time of each job with respect to the scheduling.

Let *RT* be the run time of the j-th and i-th that execute in Grid environment. We define

$$RT_j > RT_i \quad ;j > i \qquad (3)$$

Equation (3) represents the run time of each job in Grid environment and it is to order the run time of each job to the scheduling.

Equation (4) represents the average waiting time that calculates from current chromosomes and new average waiting times that calculate from new chromosomes.

$$Min(AWT_{CurrentChromosome}) \ge Min(AWT_{NewChromosome}) \qquad (4)$$

In equation (5), Let *Makespan* be the make-span time of all jobs that execute in Grid. We define

$$Min(Makespan_{CurrentChromosome}) \ge Min(Makespan_{NewChromosome}) (5)$$

The equation can be given weights $\beta, \alpha, \phi, \mu, \theta$ to obtain the fitness function F:

$$F = \sum_{j=1}^{j=n} (\beta + \alpha + \phi + \mu + \theta) \qquad (6)$$

5.4 Selection

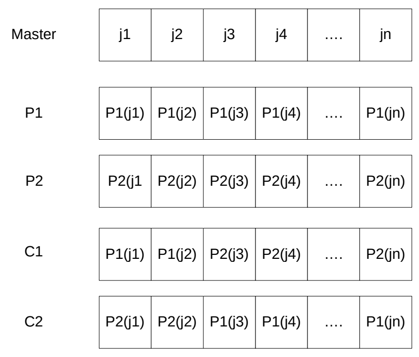
Selection models use the survival of the fittest mechanism. Fitter solutions survive while weaker ones perish. In the GA, a fitter string is more likely to receive a higher number of offspring, increasing its chances of survival.

In the proportionate selection scheme where a string with fitness value Fi is allocated a relative fitness of Fi=F, where F is the average fitness of the population. The GA uses the roulette wheel style of selection to implement proportional selection. Each string is allocated a sector (slot) of a roulette wheel with the angle subtended by the sector at the center of the wheel. The algorithm selects strings until the next generation is completely generated.

5.5 Crossover

Crossover produces new individuals that have some portions of both parent's genetic material. A crossover rate Pc is chosen to promote rapid convergence. This rate has been chosen for use in Grid scheduling through a large number of experiments. If size of populations is Pop, then Pc*Pop parent are chosen, through the selection process, to create a new population.

The crossover process can produce illegal chromosomes that can contain the duplicates of the same job(s). We use the uniform permutation crossover [21] when we generate two child chromosomes from two parent chromosomes. It has two main steps. In the first step, we randomly generate a child chromosome *(C)* from parent chromosome *(P).* Then at each position *(i)* of *C* such that $C(i) = j_i$ and $P(i) = j_i$, we copy $P_{1,2}(i \to n)$ to $C_{1,2}(i \to n)$ by randomization of position *(i)* in chromosome and we crossover times equal *Pc*Pop*. Fig. 2 shows an example of the crossover process.



Master	j1	j2	j3	j4	….	jn
P1	P1(j1)	P1(j2)	P1(j3)	P1(j4)	….	P1(jn)
P2	P2(j1	P2(j2)	P2(j3)	P2(j4)	….	P2(jn)
C1	P1(j1)	P1(j2)	P2(j3)	P2(j4)	….	P2(jn)
C2	P2(j1)	P2(j2)	P1(j3)	P1(j4)	….	P1(jn)

Figure 2

5.6 Mutation

In GA, mutation is randomly applied with a low probability, typically in the range of 0.1-1.0 percent. In the GA, mutation is a background operator, ensuring that the probability of finding the optimal solution is never zero. Mutation also acts as a safety net to recover good genetic material that may be lost through selection and crossover.

A mutation rate P_m is chosen experimental, also to promote rapid convergence. If size of the exiting population is *Pop*, then P_m * *Pop* chromosomes are chosen. For mutation process we generate randomly an integer *j* for each position *(i)* and then swap the elements at positions i and j in the chromosome. A high value of P_m can reverse the progress towards convergence; hence, this value must be selected in each case through careful study.

To make the new population size equal to the existing population size, after the processes of crossover and mutation have been used, the remaining chromosomes are selected out of the existing population by making them identical with the best existing chromosomes, chosen through the selection process. This is referred to as elitism. The following algorithm expresses the basic ideas of the GA approaches with multi-objective.

```
GA_Multi-Objective( )
{
   Generate initial population of Jobs individuals
   Evaluate individuals according to fitness
   function;
  While stopping condition is satisfied.
  {
    Count from 1 to amount generation;
    Select two parents from initial population
    (Population ⟶ Parent_1 and Parent_2);
    Crossover (Parent_1 and Parent_2) ⟶ Child;
    Mutation (Child);
   Fitness (Child);
   Improvement (Child);
   Replace (Chromosome, Child);
   Scheduling(best chromosome);
  }
 } return set of the best chromosome in population
for job scheduling.
```

6 Experimental setup and results

In this experiment, we used jobs workload data from the Standard Workload Archive [18]. This data consists of 18,239 jobs, each of which has 18 properties field, however we focused on some properties that previous mention. In our experiments we assumed that each of jobs is allowed to run in each node by using space-sharing mechanism. Our simulation, we simulated 500 different performance. nodes in Grid environment.

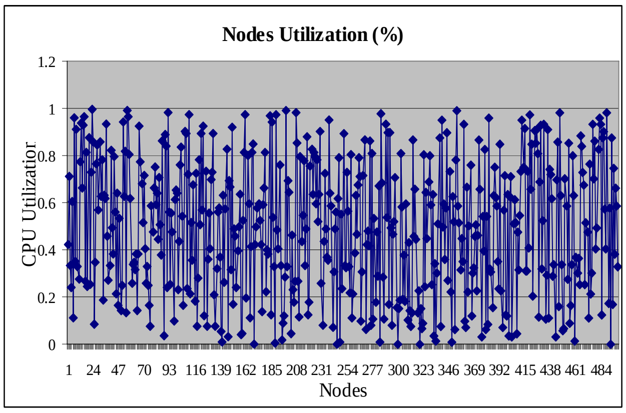


Figure 3 Nodes workload in Grid environment

Our experiments showed the average waiting time and make-span time of all jobs We define crossover rate (P_c) is 0.9, mutation rate (P_m) is 0.9 and 1,000 generation with 10,000 populations for our experiments.

Figure 4, 5 shows the average waiting time and make-span time by using genetic algorithm with multi-object.

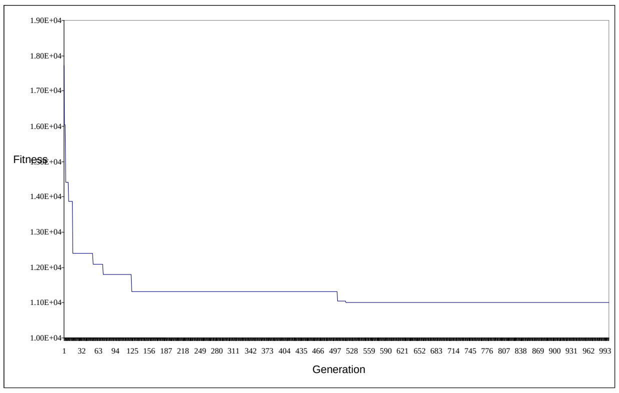


Figure 4 Average waiting time of all jobs in Grid environment

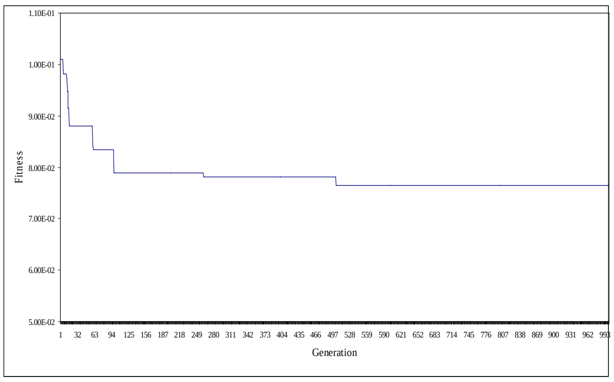


Figure 5 Make-span time of all jobs in Grid environment

7 Conclusion

We have studied the job scheduling problem for Grid environment as a combinatorial prediction and optimization. Several observations are in order.

1. We have proposed a intelligence scheduling in Grid environment and used it for our algorithm by using genetic algorithm with multi-objective.
2. We used jobs workload from the Standard Workload Archive [18] on space-sharing mechanism. We show that the proposed model captures the jobs characterization of real workload in three different classifications. This model can be used to our algorithm for simulating and evaluating scheduling policies for simulating Grid environment.
3. Many scheduling algorithms for Grid environment depend on static information provided by the Standard Workload Archive [18] and the different performance nodes in Grid, simulating by us. We observe that this information is often unreliable. However, it is a useful way.
4. The experiment results on the waiting time and the makespan have shown that the scheduling using our algorithm can allocate the best results.
5. The results provided here suggest that the researcher look forward to new method for such problems should consider combine them with their method.

8 Future Work

We conclude the interesting topic problems are in order.

1. Our simulation environment will include critical parameters, such as submit time, Grid network cost, job migrations overhead, fault tolerance.
2. We plan to investigate the swarm intelligence mechanism, such as ant colony algorithm for Grid scheduling.
3. We will include a more complex characterization of the constraints for Grid scheduling and will improve the complexity problems in Grid environment.

9 Acknowledgements

We would like to thank Suan Dusit Rajabhat University for supported us.

References:

[1] Foster I, Kesselman C, Tuecke S. The anatomy of the Grid: Enabling scalable virtual

organizations. International Journal of Supercomputer Applications 2001.

[2] I. Foster and C. Kesselman, Eds., The Grid 2: Blueprint for a New Computing Infrastructure. San Francisco, CA: Morgan Kaufmann, 2004.

[3] M. Parashar, J. Browne, Conceptual and Implementation Models for the Grid.

[4] Global Grid Forum [Online]. Available: http://www.gridforum.org.

[5] ArshadAli, Ashiq Anjum, Julian Bunn, R. Cavanaugh, Frank van Lingen, R. McClatchey, Muhammad Atif Mehmood, H. Newman, C. Steenberg, M. Thomas, I. Willers. PREDICTING THE RESOURCE REQUIREMENTS OF A JOB SUBMISSION.

[6] Y. Gao, H. Rong, Joshua Zhexue Huang. Adaptive grid job scheduling with genetic algorithms.

[7] V. Di Martino, M. Mililotti, Schduling in a Grid computing environment using Genetic Algorithms, Proceeding of the International Parallel and Distributed Processing Symposium, 2002.

[8] M. Aggarwal, Robert D., Kent and Alioune Ngom, Genetic Algorithm Based Scheduler for Computational Grids, Proceeding of the International Symposium on High Performance Computing System and Applications, 2005.

[9] V. Di Martino, M. Mililotti, Sub optimal scheduling in a grid using genetic algorithms, Parallel computing, 2004.

[10] U. Fissgus, Scheduling Using Genetic Algorithms.

[11] Y. Zhang, H. Ranke, J.E. Moreira, A. Sivasubramaniam, An integrated approach to parallel scheduling using gang-scheduling, backfilling and migration, in: Lecture Notes in Computer Science, Vol. 2221, Springer, Berlin, 2001, pp. 133-158.

[12] O. Beaumont, A. Legrand, Y. Robert, Optimal algorithms for scheduling divisible workloads on heterogeneous systems, in: Proceedings of the International Parallel and Distributed Processing Symposium, 2003.

[13] H. Casanova, A. Legrand, D. Zagorodnov, F. Berman, Heuristics for scheduling parameter sweep applications in Grid environments, in: Heterogeneous ComputingWorkshop 2000, IEEE Computer Society Press, 2000, pp. 349–363.

[14] Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/, Juni 2004.

[15] J. Jann, P. Pattnaik, H. Franke, F. Wang, J. Skovira, and J. Riordan. Modeling of Workload in MPPs. In D. Feitelson and L. Rudolph, editors, IPPS'97 Workshop: Job Scheduling Strategies for Parallel Processing, pages 94–116. Springer–Verlag, Lecture Notes in Computer Science LNCS 1291, 1997.

[16] C. Ernemann, B. Song, and R. Yahyapour. Scaling of Workload Traces. In D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, Job Scheduling Strategies for Parallel Processing: 9th International Workshop, JSSPP 2003 Seattle, WA, USA, June 24, 2003, volume 2862 of Lecture Notes in Computer Science (LNCS), pages 166–183. Springer-Verlag Heidelberg, October 2003.

[17] Thomas L. Casavant, Taxonomy of Scheduling in General-Purpose Distributed Computing Systems, IEEE Transactions on Software Engineer, Februart,1988.

[18] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, Mass.: Addison-Wesley, 1989.

[19] J.J. Holland, Adaptation in Natural and Artificial Systems. Ann Arbor, Mich.: Univ. of Michigan Press, 1975.

[20] M. Negnevitsky, Artificial Intelligence: A guide to intelligent system, Addison-Wesley, 2002.

[21] Goldberg D.E, Genetic Algorithm in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA. 1989.