Multi-Objective Job Scheduler using Genetic Algorithm in Grid Computing

Pritibahen Sumanbhai Patel
Asst. Prof., CVR College of Engg.
Vastunagar, Mangalpally
Hyderabad, Andhra Pradesh, India

ABSTRACT
This paper presents multi-objective Job scheduler using Genetic Algorithm which provides efficient utilization of resources by completing the different tasks in a minimum period of time. Grid is a kind of distributed system that provides the sharing of geographically distributed independent resources dynamically at runtime depending on their availability, capability, performance and cost. Scheduling is a key problem in evolving grid computational systems. Dealing with the multiple criteria in a heterogeneous and dynamic environment like Grid is very complex and computationally hard. There are ample approaches for Job scheduling like Genetic Algorithm (GA), Simulated Annealing (SA), Ant Colony optimization (ACO) and Particle Swarm Optimization (PSO) Algorithm. This paper presents Genetic algorithm for designing efficient multi-objective job schedulers by considering multiple parameter like makespan and flow time to find optimal/nearly optimal schedule. It searches solution space in parallel and solution can be found more quickly.

General Terms
Grid computing, Genetic Algorithm, Scheduler.

Keywords
Genetic Algorithm (GA), Scheduler, Makespan, Minimum completion time, Fitness, Flow Time.

1. INTRODUCTION
The computational grid is dynamic and heterogeneous in nature. It is focusing on large-scale resource sharing, innovative applications and high-performance orientation. In grid resources may vary in performance and capacity. The resources may leave or join the grid at any time. Hence, the scheduling is important in the grid. So GAs for designing efficient multi-objective Grid schedulers when makespan and flowtime are minimized is discussed. The GA operation is based on the Darwinian principle of "survival of the fittest". It implies that the fitter individuals are more likely to survive and have a greater chance of passing their good genetic features to the next generation. In genetic algorithm, each individual that is a member of the population represents a potential solution to the problem. GA starts with initial population of individuals (chromosomes). Each individual is evaluated using fitness function to produce a value known as goodness of the solution. Then a new population is generated by selecting best individuals from the current population and applying crossover operator to produce new offspring which would inherit good features of parents. Then each offspring is mutated in order to prevent GA to be trapped in local optima. Best individuals among current population and new population are carried forwarded in the next generation. The process is repeated until stopping condition met and best solution in the current generation is returned.

2. LITERATURE SURVEY
The existing approach for grid jobs scheduler implemented with conventional algorithm techniques which may give optimal solution but not in reasonable amount of time. The survey shows several limitations like algorithms are studied using simulation, mostly static in nature, do not react to dynamism involved in the grid environment, studied for small sized problems only. This paper presents implementation of multi-objective job scheduler which will address all of the above problems. It is based on Genetic algorithms which gives optimal/nearly optimal solution quickly.

3. PROPOSED SYSTEM DESIGN
Here multi-objective jobs scheduler using GA is implemented to find optimal/nearly optimal schedule when makespan and flow time are minimum, which efficiently utilize the resources. Proposed multiparameter jobs scheduler using GA can quickly search solution space in parallel to find optimal/nearly optimal solution in very less time. It uses dynamic information received from Grid Information System to determine optimal/ nearly optimal solution. It can work with larger sized problems.

4. PROBLEM FORMULATION
Multiparameter jobs scheduler using GA is based on Expected Time to Compute (ETC) Model. An ETC for any job j on any resource (machine) r is expected execution time of job j on r if j is scheduled on r. The problem for grid scheduling consists of following:

- n – the number of jobs to be schedule at particular instance of time. Any job has to be processed entirely in unique resource.
- m – the number of heterogeneous resources(machines) available in the Grid for an execution of a given set of jobs
- N = {j1, . . . ,jn} a set of n jobs
- R = {r1, . . . ,rm} set of available m resources.
- The workload Wi of each job i.
- The computing capacity CCr of each resource (in millions of instruction per second) r.
- The expected time to compute ETC matrix of size n×m (number of jobs * number of resources). ETC[j][r] indicate the expected execution time of job j in resource r.

This paper consideres the scenario in which jobs submitted to the Grid are independent and are not preemptive.

4.1 Fitness of a Schedule

The first criterion for getting optimal/nearly optimal schedule is minimizing the makespan and the secondary criterion is to minimize the flowtime. These two criteria are defined as follows.

4.1.1 Makespan

The time when latest job finishes. It is calculated as follows:

$$\text{makespan} = \min_{S_i \in \text{Schedules}} \left\{ \max_{j \in N} F_j \right\} \dots\dots\dots\dots\dots\dots\dots (1)$$

4.2.2 Flowtime

The sums of finalization time all the jobs. It is calculated as follows:

$$\text{flowtime} = \min_{S_i \in \text{Schedules}} \left\{ \sum_{j \in N} F_j \right\} \dots\dots\dots\dots\dots\dots\dots (2)$$

In eq.(1) & (2) Fj denotes time when job j finalizes, schedules denotes the set of all possible schedules and N denotes the set of all jobs to be scheduled. Makespan is not affected by any particular execution order of the jobs in a concrete resource. But to minimize flowtime, the jobs need to be considered in ascending order of their Expected Time to Compute (ETC). It should also be noted that makespan and flowtime are contradictory objectives. Trying to minimize one of them could not suit the other. In order to express the Eq. (1) in an easily computable form, makespan is expressed in terms of the completion time of a resource (machine). Let completion be a vector of completion times of m resources. The completion time of machine i is denoted by completion[r] and it is expressed as a total time needed for the resource r to finalizing its previously assigned jobs and jobs which are actually scheduled to this resource. Completion time of machine i is denoted by completion[r] and it is expressed as a total time needed for the resource r to finalizing its previously assigned jobs and jobs which are actually scheduled to this resource. Computation of ETC and completion time (completion[r]) for resource r as follows:

$$ETC[j][r] = \frac{W_j}{CC_r} \dots\dots\dots\dots\dots\dots\dots (3)$$

$$\text{completion}[r] = \text{ready}_r + \sum_{\{j \in N | \text{Schedules}[j] == r\}} ETC[j][r] \dots\dots\dots\dots\dots\dots (4)$$

Where,

ETC[r][j]=expected time to compute job j on resource r.

W_j=workload of job j

CC_r=computing capacity of resource r.

completion[r]=completion time for resource r.

$ready_r$=time when resource r finishes previously assigned jobs to it.

The makespan of eq.(1) can be redefined as the maximal completion time and can be calculated as follows:

$$\text{makespan} = \max\{\text{completion}[r] | \ r \in R\} \dots\dots\dots\dots\dots\dots (5)$$

The flowtime in eq.(2) can be redefined as follows:

$$ft[r] = \text{ready}_r + \sum_{\{j \in \text{Sorted}(r)\}} ETC[j][r] \dots\dots\dots\dots\dots\dots (6)$$

$$\text{flowtime} = \min\{ft[r] \mid r \in R\} \dots\dots\dots\dots\dots\dots (7)$$

Where,

ft[r]=flowtime for machine r.

Sorted(r)= the set tasks assigned to the machine i sorted in ascending order by the corresponding ETC values.

These two criteria makespan and flowtime can be integrated in several ways to establish the desired priority among them. For optimization two fundamental models are the hierarchical and the simultaneous approach. In hierarchical approach, the optimization criteria are sorted by their importance. The process starts by optimizing most important criterion. When further improvements are not possible, the second criterion is optimized while keeping optimized value of first important criterion unchanged. In grid scheduling, makespan may be considered as most important criterion and flowtime may be secondary criterion.

In the simultaneous approach, makespan and flowtime are minimized simultaneously. This paper uses weighted sum function since makespan and flowtime are measured in same time unit. However, the flowtime values are usually higher magnitude than makespan. For this reason, the value of mean flowtime is computed. Moreover, both values are weighted in order to balance their importance. This paper uses simultaneous approach to compute objective function or fitness function.

$$\text{mean\_flowtime} = \frac{\text{flowtime}}{R} \dots\dots\dots\dots\dots\dots (8)$$

$$\text{fitness} = \frac{\gamma}{\text{makespan}} + \frac{1-\gamma}{\text{mean\_flowtime}} \dots\dots\dots\dots\dots\dots (9)$$

Where γ has been fixed by initial tuning process.

5. OVERALL SYSTEM ARCHITECURE

This section presents a GA based grid multi-objective job scheduler that maximizes resource utilization by minimizing makespan and flowtime. It also determines schedules based on the current resource information. And hence can easily react to dynamism involved in grid environment. Overall system architecture shown in figure 1. System is designed in 3 major modules.

5.1 Monitoring & Discovery Service (MDS) Module

This module discovers the new grid resources and monitors already discovered resources. When MDS process starts first time it reads /var/grid resources file to get list of the resources available initially. It also creates a thread to periodically poll already discovered resources to get current information about each of these grid resources. The information includes static information like processor family/architecture, number of CPUs/resource, CPU frequency, total RAM, total swap area etc., and dynamic information such as resource computing status busy/free, resource up/down status, free RAM, load, number of free CPUs etc. It also periodically receives resource information from grid resources. This information is sent to manager process as well as GA based grid scheduler as and when needed. GA scheduler uses current resource information to compute optimal/nearly optimal solution to assign jobs to resources. It also receives update information from manager process and updates its data structures accordingly.

5.1 Manager Module

This module is the central part of implementation. It receives

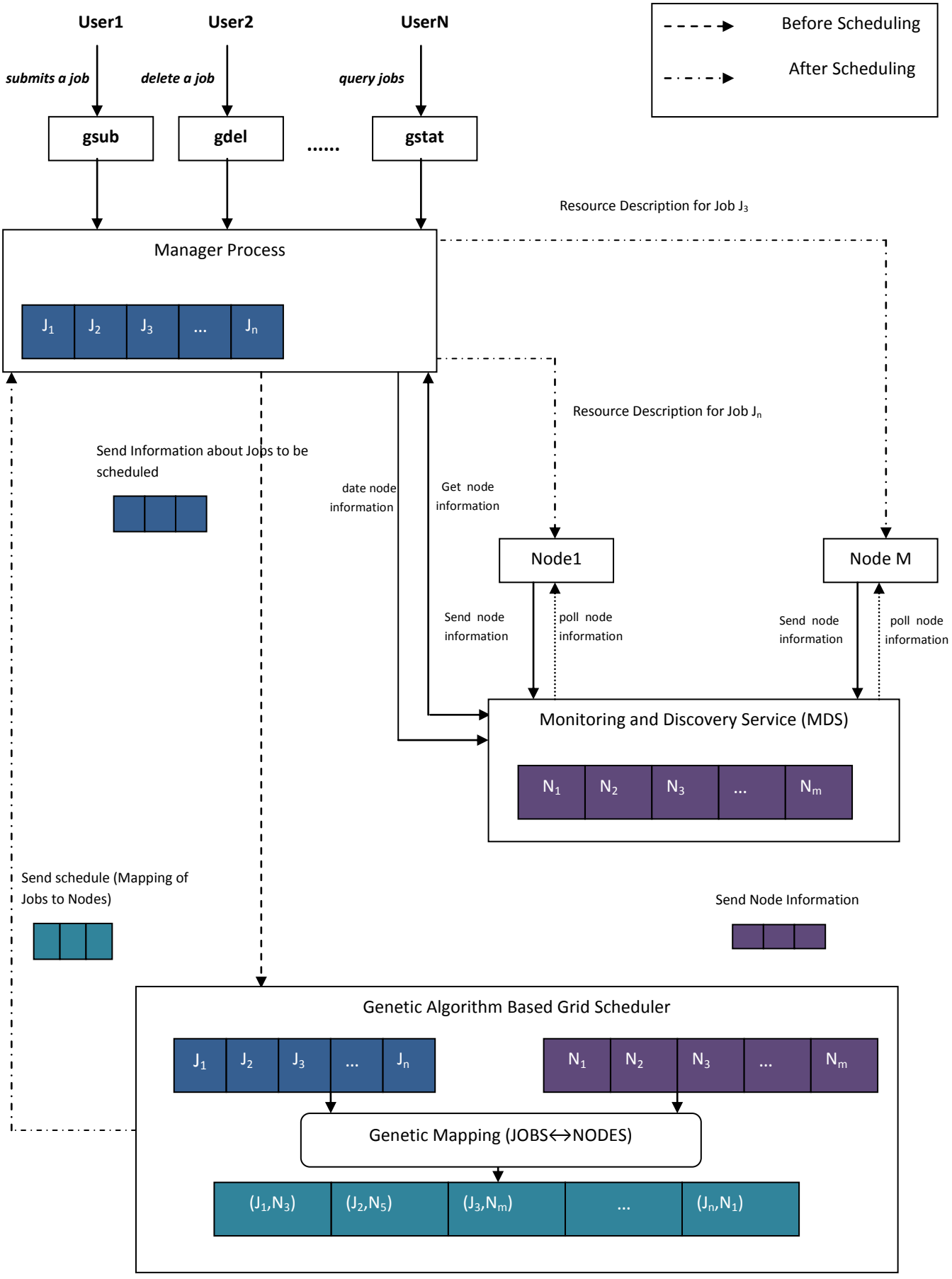

Figure 1: Proposed system design

commands from users. It implements following functions.

- Command processing & Scheduler invocation
- Job queue management
- Job management
- Job monitoring

The Manager receives command requests (such as submit a job, query jobs, delete a job) from users as shown in figure 1. When a user submits a job using gsub command, it sends job submission request to manager. When manager receives a job submission request, unique job id is generated for a job and its description is appended to job queue. If a command request is to query jobs(gstat), it simply loop through job queue and send information such as job id, job status, job name, job executable, assigned resource if it is already scheduled etc. If command is to delete a job (gdel) and job is scheduled then job management components forward request to gatekeeper of the assigned host to clean the job. Once the job is deleted on the resource, it will be removed from the job queue otherwise an error is reported. This component periodically checks if there are unscheduled jobs in the job queue. If there are some jobs, it connects to GA Grid scheduler, send information about jobs to GA grid scheduler and wait for optimal/nearly optimal mapping of jobs to suitable resources from the scheduler. Once it receives, a optimal/nearly optimal schedule from scheduler, for each (job, resource) pair in the schedule, it generates Globus GRAM description for the job. It submits this GRAM job description to the assigned resource using Globus command globus-job-submit.. It also periodically monitors status of the scheduled jobs using Globus command globus-job-status and updates job information in the job queues. To delete already scheduled jobs, it uses Globus command globus-job-clean. The detailed job submission procedure is shown in figure 2.

5.2 Scheduler Module

This module uses GA to find optimal/nearly optimal solution by minimizing makespan and flowtime. It receives information about list of jobs from manager and information about available resources from MDS server. It then creates initial population of k schedules using Minimum Completion Time (MCT) heuristics. It then evaluates the current population by computing fitness function for each of k. It then creates a new population by repeating selection, crossover, and mutation and assignment steps until the size of new population becomes k. It then evaluates the new population and carries forward best schedules of the current population as well as the new population in the next generation in order to get optimal/nearly optimal solution quickly. The algorithm evolves generation by generation until termination criteria met. The scheduler then returns best schedules in current population. This schedule will then be sent to manager. Manager submits this job description to the assigned resource.

6. SYSTEM DESIGN

This section presents actual design of Multi-objective Job scheduler using Genetic Algorithm in grid computing.

6.1 Schedule encoding

This section presents direct representation to encode each possible schedule in a chromosome. There is an array chromosome of n(number of jobs) integer to represent a chromosome(a schedule) as shown in figure 3. Chromosome[j] represents the resource number where job j is scheduled.

6.2 Generation of Initial population

In GA, initial population is usually generated randomly. But

Job No:	1	2	3	4	5	6	7
Resource No:	4	2	7	6	3	5	1

Figure 3: Encoding of a schedule (a chromosome)

to guide the searching process and to get optimal/nearly optimal solution in fewer generations with the help of several problem specific heuristics such as Min-Min, Minimum Completion Time (MCT) etc. This paper presents use of MCT heuristics to guide a searching process for finding optimal/nearly optimal schedule quickly in fewer generations. In the MCT heuristic, each job is assigned to the resource where job completes in minimum time. Jobs are considered for allocation at random.

6.2.1 Compute Fitness function

The scheduler aims to maximize resource utilization by minimizing makespan and flowtime. Good chromosomes have higher fitness values. The fitness of each chromosome (schedule) is computed using eq.(9).

6.2.2 Selection operator

Selection operator is used to select parents to which crossover operator is applied to produce new offspring. In general, selection is directly proportional to the fitness of chromosomes. Several selection methods exist to select chromosomes for crossover such as linear ranking, roulette wheel selection etc. Here roulette wheel selection technique is used to select good schedules to produce new offspring. In roulette wheel selection method, the probability that a chromosome selected is directly proportional to its fitness value. Higher the fitness, higher chances the chromosome will be selected. In this method, each schedule or chromosome gets portion on the roulette wheel according to its fitness value. Chromosomes with higher fitness value get larger slice on roulette wheel. Selection is done by spinning a roulette wheel. Since fittest schedule has larger portion on the roulette wheel, they will have higher chance of being selected. Circumference of roulette wheel represents the total fitness of all chromosomes. Pseudo code for roulette wheel selection method is shown in figure 4. The roulette wheel selection of among 4 chromosomes is shown in figure 5. Chromosome 3 has higher chance of getting selected as shown in figure 5.

```
RouletteWheelSelection()
{
    total_fitness=0.0;
    running_sum=0.0;
    for each chromosome k in a current population
```

```
        total_fitness=fitness(k);
    r=select random number r in the range [0,total_fitness-1];
     for each chromosome k in a current population
          running_sum=running_sum+fitness(k);
          if(running_sum >= r)
              return(k);
}
```

Figure 4: Pseudo code Roulette Wheel Selection

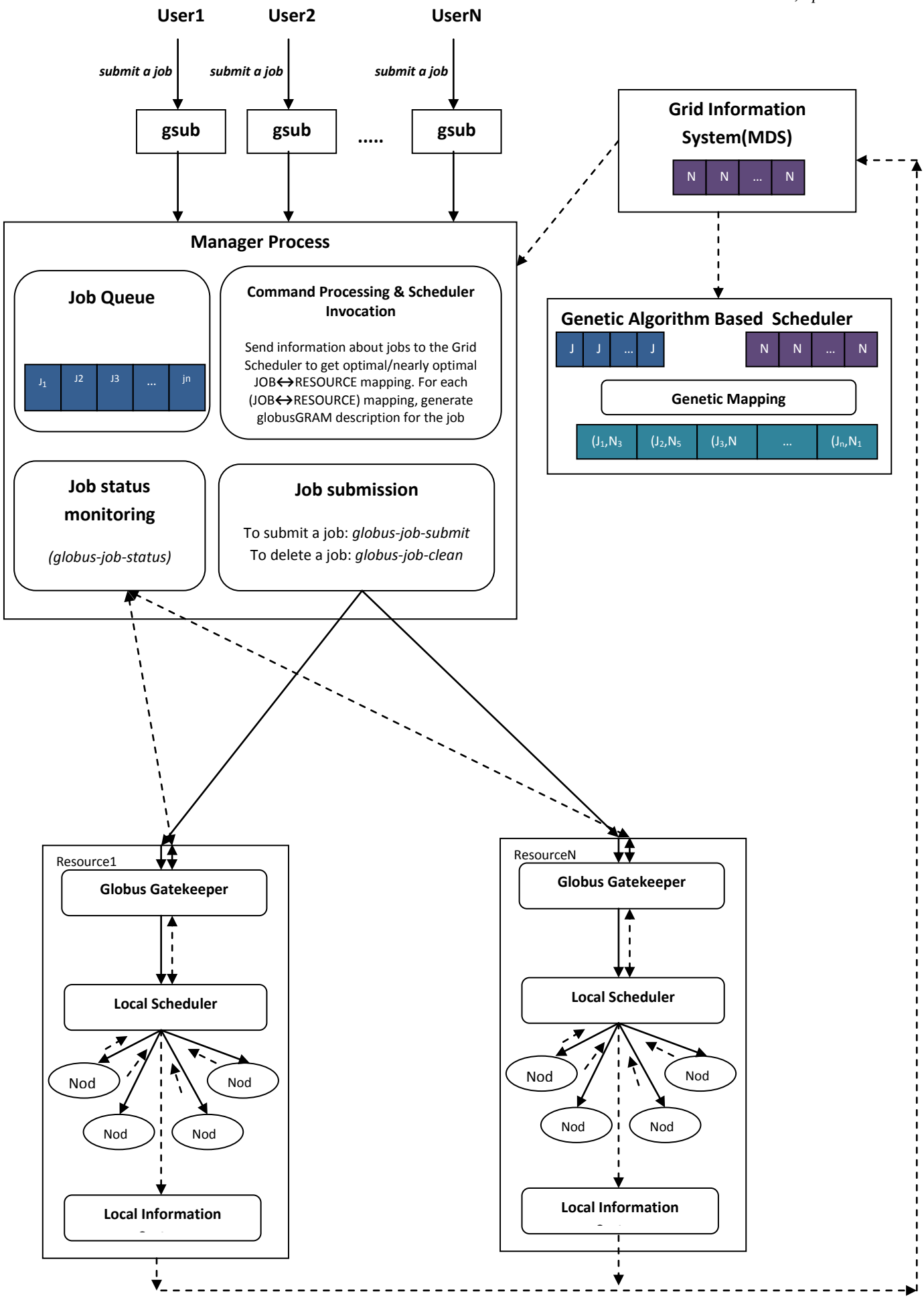

Figure 2: Detail flow diagram for job submission

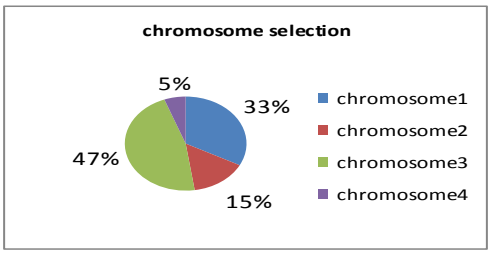

Figure 5: Roulette wheel selection among 4 chromosomes.

6.2.3 Crossover operator

With crossover operator, two selected parent chromosomes can interchange their genes and produce new offspring (children). The aim is to obtain better quality solution and explore a new region of solution space that has not been yet explored. One may use several different types of crossover such as one-point crossover, two-point crossover, uniform crossover etc. Here one-point crossover operator method is used to produce offspring schedules. In this method, first, random crossover point between 1 and n(number of jobs) is selected, and then first parts of two parents are interchanged to produce two offspring(schedules). Same way, exchanging second parts of two parents to produce two new offspring (schedules) which are same as those produced by exchanging first parts. One point crossover is explained in figure 6.

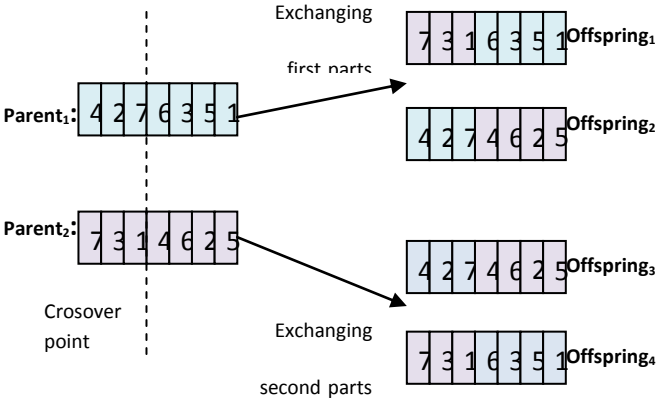

Figure 6: One-point crossover operation to produce 4 offspring schedules

6.2.4 Mutation operator

Mutation randomly changes gene(s) to different values. It is used to provide diversification by changing some gene(s) randomly and thereby prevent GA search process getting stuck in to local optima. There are several types of mutation such as move, swap etc., applied to a schedule. Here move mutation is used to randomly selects a job in a schedule (a chromosome) and assign it to another machine as shown in figure 7.

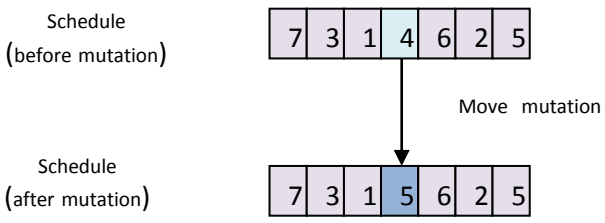

Figure 7: Mutation operation

6.2.5 Replacement operator

Replacement operator determines which of the chromosomes (schedules) survives in the next generation. Two kinds of replacement usually used to carry forwards chromosomes to next generation (a) Generational replacement (b) Partial replacement. In a generational replacement, the current population is entirely replaced by new population while in partial replacement worst chromosomes in a current population are replaced by good chromosomes of new population. This paper presents use of partial replacement strategy in which k best chromosomes from combined current and new population are carried forward to the next generation. First, fitness function is computed for each offspring. Let CP(t) be the current population in generation t and NP(t) be new population in generation t, then current population of next generation t+1 will be

CP(t+1)=k best schedules from (CP(t) U NP(t))

6.2.6 Termination Criteria

Termination criteria could be:

(i) Maximum number of generations or iterations: the genetic search process is terminated after fixed number of generation.

(ii) Number of iterations without improvement: the optimization process is terminated after some fixed number of iterations without any improvement.

Here (i) termination criterion is used for GA based scheduler in which search process terminates after 300 generations. If termination criterion is not satisfied goto step 3 and repeat the process.

In general, this genetic search process can be summarized as follows:

GAGridScheduling() {

ENCODING: Represent a schedule(a chromosome) using array of n(numof jobs) integer chromosome such that chromosome[i] represents the resource on which job is scheduled

INITIALIZATION: Generate a initial population CP(t=0) of k schedules using MCT(Minimum Completion Time) heuristic.

FITNESS: Evaluate schedule in CP(t) using eq. (5)

TERMINATION CRITERIA: Check if termination criteria satisfied, if 'yes' return the best solution from current population CP(t).

NEW POPULATION: Repeat following steps until size of new population NP(t) becomes k.

Selection: Select two parents schedules p1 & p2 from CP(t) using roulette wheel method.

Crossover: With crossover probability pc perform one-point crossover to produce two new offspring schedules o1 & o2.

Mutation: With very low mutation probability pm, change the assignment of randomly chosen job to new grid resources in each offspring o1 and o2.

Assignment: Place o1 & o2 in NP(t)

NP(t)=NP(t) U{ o1,o2}

FITNESS: Evaluate schedule in NP(t) using eq.(5).

REPLACEMENT:

Select k best schedules from CP(t) and NP(t) to carry forward in the next generation. CP(t+1)=k best schedules from (CP(t) U NP(t))

Increment generation count

t=t+1

Goto Step 4

}

7. RESULTS & ANALYSIS

For the experimental purpose consider following problem instance consisting of 10 grid resources and 20 jobs. List of grid resources with existing workload is shown in the table-1.

Table-1: List of grid resources with corresponding computing capacity

Job No	Workload
1	126
2	233
3	759
4	858
5	829
6	255
7	789
8	898
9	547
10	110
11	595
12	394
13	582
14	394
15	908
16	310
17	568
18	530
19	125
20	804

To find out optimal/nearly optimal solution for this problem instance, GA based multi-objective scheduler with following parameters is shown:

Number of Generations=300

Size of population=256

Crossover probability(Pc)=0.90

Mutation probability(Pm)=0.0001

got makespan=26.0183 in generation number 189 and then it retains this value until last generation. So if number of generations is reduced to less then 189, makespan is 26.6066. The graph of generation numbers vs makespan for this problem is shown in figure 8 where Y-axis represents makespan values and X-axis represents generation number.

Table-2: List of jobs with corresponding workload

Resource No.	Computing Capacity (MIPS)	Existing workload (pending processing in ms)
1	3380	72.88
2	931	43.44
3	2969	69.92
4	3120	97.47
5	3728	47.61
6	1815	32.22
7	3170	22.67
8	2084	46.86
9	2014	26.48
10	3318	46.09

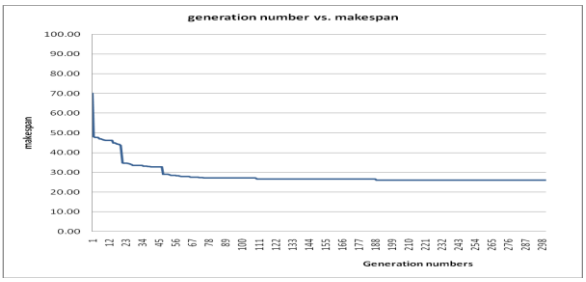

Figure 8: Makespan vs. generation numbers

8. CONCLUSION

Genetic Algorithms (GAs) for designing efficient Grid schedulers when makespan and flowtime parameters are minimized under simultaneous approaches is shown. The experimental study reveals the quality of the proposed GA-based multi-objective schedulers as compared well to the existing GA-schedulers in the literature. This GA-based schedulers can be used to design dynamic schedulers. A dynamic scheduler would run GA in batch mode to schedule jobs arrived in the system since last activation of the scheduler.

9. FUTURE SCOPE

Future work will show GA scheduling based on workflow scheduling. Workflow management system allows the user to specify their requirements along with the descriptions of tasks and their dependencies using the workflow specification. It will be integrated with various grid middleware such as UNICORE, LIGEON etc. Study of implementing same scheduler with different heuristics such as min-min, max-min, MET etc. Extension of scheduler for multi matches maker between user's requirement and resource characteristics.

10. REFERENCES

[1] Javier Carretero, Fatos Xhafa, Ajith Abraham. Genetic algorithm based schedulers for grid computing systems. In International Journal of Innovative Computing, Information and Control ICIC International °c 2005 ISSN 1349-4198 Volume 3, Number 5, October 2007.

[2] Jing Liu, Li Chen, Yuqing Dun, Lingmin Liu, Ganggang Dong. The Research of Ant Colony and Genetic Algorithm in Grid Task Scheduling. In International Conference on MultiMedia and Information Technology 2008.

[3] S. Prabhu, V.Naveen Kumar. Optimization Based on Genetic Algorithm in Grid Scheduling. International Journal of advanced research in technology. IJART, Vol. 1 Issue 1, 2011. ISSN NO: 6602 3127 RR.

[4] Abraham, A. H. Liu, W. Zhang and T. G. Chang, Job scheduling on computational grids using fuzzy particle swarm algorithm, Proc. of the 10th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, B. Gabrys et al. (eds.): Part II, Lecture Notes on Artificial Intelligence 4252, 500507, Springer, 2006.

[5] Jia Yu and Rajkumar Buyya and Kotagiri Ramamohanarao. Workflow Schdeduling Algorithms for Grid Computing. Grid Computing and Distributed Systems (GRIDS) Laboratory Department of Computer Science and Software Engineering, The University of Melbourne, Australia.

[6] Guangchang Ye, Ruonan Rao, Minglu Li. A Multiobjective Resources Scheduling Approach Based on Genetic Algorithms in Grid Environment. In Fifth International Conference on Grid and Cooperative Computing Workshops (GCCW'06) IEEE computer society.

[7] Taras S. Shapovalov, Alexey G. Tarasov. Genetic Algorithm Based Parallel Jobs Scheduling. In program "Research and scientific-pedagogical personnel of innovative Russia"(project No. 02-740-11-0626) and Grant of Russian Foundation for Basic Research and Far eastern branch of Russian academy of sciences No. 10-III-B- 01I-009.

[8] Wei Sun , Yuanyuan Zhang , Yanwei Wu, and Yasushi Inoguchi. Practical Task Flow Scheduling for High Throughput Computational Grid. In International Conference on Parallel Processing Workshops (ICPPW'06) 0-7695-2637-3/06, 2006,IEEE computer society.

[9] A. Abraham, R. Buyya, and B. Nath. Nature's heuristics for scheduling jobs on computational grids. In The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), India, 2000.

[10] Arash Ghorbannia Delavar, Mohsen Nejadkheirallah, Mehdi Motalleb. A New Scheduling Algorithm for Dynamic Task and Fault Tolerant in Heterogeneous Grid Systems Using Genetic Algorithm. In IEEE computer society 2010.

[11] Dr. K.Vivekanandan, D.Ramyachitra A Study on Scheduling in Grid Environment Dr. K. Vivekanandan et al. / International Journal on Computer Science and Engineering (IJCSE).

[12] Javier Carretero, Fatos Xhafa. Use of Genetic algorithm for scheduling jobd in large scale grid applications. In Okio Technologies IR Ekonominis Vystymas Technological and Economic Development of Economy, ISSN 1392-8619 Volume XII, Number 1, 2006.

[13] Wael Abdulal, Omar AI Jadaan, Ahmad Jabas, S. Ramachandram. An Improved Rank-based Genetic algorithm with limited Iterations for grid Scheduling. In IEEE symposium on Industrial Electronics and Applications(ISIEA 2009), Kaula Lumpur, Malaysia, October 4-6, 2009

[14] Weizhe Zhang, Albert M.K. Cheng, Mingzeng Hu. Multisite Co-allocation Algorithms for Computational Grid. In IEEE , 2006

[15] Suchang Guo, Hong-Zhong Huang, Zhonglai Wang, Min Xie. Grid Service Reliability Modeling and Optimal Task Scheduling Considering Fault Recovery. In IEEE Transactions on reliability, VOL.60, NO.1, March 2011.

[16] Vijay Subramani, Rajkumar Kettimuthu, Srividya Srinivasan, P.Sadayappan. Distributed Job Scheduling on Computational Grids using Multiple Simultaneous Requests*

[17] Ajith Abraham, Hongbo Liu, Crina Grosan, Fatos Xhafa. Nature Inspired Meta-heuristics for Grid Scheduling: Single and Muti-objecive Optimization Approaches. F. Xhafa, A. Abraham(Eds.):Meta. For Sched. In Distri. Comp. Envi.,SCI 146, pp. 247-272, 2008, Springer-verlag berlin Heidelberg 2008.

[18] Wael Abdulal, S. Ramachandram. Reliability-Aware Genetic Scheduling Algorithm in Grid Environment. In IEEE International Conference on Communication Systems and Network Technologies DOI 10.1109,2011,145.

[19] Carsten Ernemann, Volker Hamscher, Uwe Schwiegelshohn, Ramin Yahyapour. On Advantages of Grid Computing for Parallel Job Scheduling, In Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID.02).

[20] S. Bhaghavathi Priya, M. Prakash, Dr. K.K.Dhawan. Fault Tolerance-Genetic Algorithm gor Grid Task Scheduling using Check Point. In IEEE the sixth international conference on grid and cooperative computing(GCC),2007.

[21] J. Monroy, J.A. Becerra, F.Bellas, R.J. Duro. Parallel Job Scheduling through Evolutionary Based Cognitive Strategies, In IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, July 16-21, 2006.

[22] Hamed Vahdat-Nej ad, Reza Monsefi, Mahmoud Naghibzadeh. A New Fuzzy Algorithm for Global Job

Scheduling in Multiclusters and Grid, In IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA), Ostuni - Italy, 27-29 June 2007

[23] Pavel Fibich and Lud˘ek Matyska and Hana Rudov´a. Model of Grid Scheduling Problem, American Association for Artificial Intelligence,2005

[24] Kamaljit Kaur, Amit Chhabra, Gurvinder Singh. Heuristics Based Genetic Algorithm for Scheduling Static Tasks in Homogeneous Parallel System, In International Journal of Computer Science and Security (IJCSS), Volume (4): Issue (2).