Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. III (2008), No. 3, pp. 249-258

A Hyper-Heuristic Approach for Efficient Resource Scheduling in Grid

S. Mary Saira Bhanu, N.P. Gopalan

Abstract: Efficient execution of computations in grid can require mapping of tasks to processors whose performance is both irregular and time varying because of dynamic nature. The task of mapping jobs to the available computing nodes or scheduling of the jobs on the grid is a NP complete problem. The NP-hard problem is often solved using heuristics techniques. Heuristic and metaheuristic approaches tend to be knowledge rich, requiring substantial expertise in both the problem domain and appropriate heuristics techniques. To alleviate this problem the concept of Hyper-heuristic was introduced. They operate on the search space of heuristics instead of candidate solutions and can be applied to any optimization problem. This paper emphasizes the use of Hyper-heuristics built on top of hybridized Metaheuristics to efficiently and effectively schedule jobs onto available resources in a grid environment thus resulting in an optimal schedule with minimum makespan.
Keywords: grid, hyper-heuristics, scheduling

1 Introduction

As the number of networked computers worldwide increases complex computations are distributed over multiple computer to improve the utilization of the computers and to meet the increasing computational requirements of scientific research. Grid computing which is an extension of distributed computing is a highly dynamic one, in which the resources join and leave the grid in an unpredictable fashion. The grid operates in a peer-to-peer fashion where resources join or leave without any pre-arranged schedule. The true computational capability is also hard to obtain at any instance of time as these computational resources can go offline regardless of the job state allocated.
The Resource management system (RMS)[6] of the grid has to select the appropriate resource and to provide best QOS. Foster [8] defines the responsibilities of the RMS on the grid as the discovery of available resources to the application, mapping the resources to the application subject to some performance goals and scheduling policies and loading the application to the resource in accordance with the best available schedule. The task of mapping jobs to the available computing nodes or scheduling of the jobs on the grid is a NP complete problem. The schedule strategies can have a significant impact on the performance characteristics. The NP-hard problem is often solved using heuristics techniques. Heuristics are intended to gain computational performance or conceptual simplicity, potentially at the cost of accuracy or precision. Simple heuristics methods may not find an optimal solution, since they perform moves that lead to a final solution, which is a local optimum. To escape from the local optima traps, metaheuristic [2] is used. A metaheuristic is an iterative master process that guides and modifies the operations of subordinate heuristics in order to efficiently produce high quality solutions. Metaheuristics operate at a higher level, over heuristics, guiding them to an optimal solution. Common metaheuristics are local search, simulated annealing, ant colony optimization, tabu search, genetic algorithms, and swarm intelligence. Many variants and hybrids of these techniques exist and can be applied in various real world applications.
Heuristic and metaheuristic approaches tend to be knowledge rich, requiring substantial expertise in both the problem domain and appropriate heuristics techniques. To alleviate this problem the concept of Hyper-heuristic [7, 13] was introduced. It is a heuristic that operates at a higher level of abstraction than the current metaheuristics approaches.
Metaheuristics are problem-specific solution methods, which require knowledge and experience about the problem domain and properties and require fine-tuning of parameters. They provide state-of-the art

solutions. Hyper-heuristics, on the other hand are developed to be general optimization methods, which can be applied to any optimization problem easily. They operate on the search space of heuristics instead of candidate solutions. Thus unlike metaheuristics, Hyper-heuristic methods deploy a set of simple heuristics and use only nonproblem-specific data, such as, fitness change or heuristic execution time. Hyper-heuristic approach is used to solve the various scheduling problems like Sales Summit problem, personnel scheduling, nurse rostering problem, course and timetabling problem. Hyper-heuristics algorithms are more adaptive to the Grid scenarios where both resources and applications are highly diverse and dynamic in nature. In this paper, we use the hyperheuristic approach to schedule the independent jobs in a grid environment.

2 Related Work

Tremendous amount of research has been going to devise efficient algorithms to schedule resources efficiently in dynamic work environments. Rafael A. Moreno [1] addresses the issues that the resource broker has to tackle like resource discovery and selection, job scheduling, job monitoring and migration etc. Lingyun Yang [14]proposed a conservative scheduling policy that uses information about expected future variance in resource capabilities to produce more efficient data mapping decisions. David Beasley,Marek Mika and Grzegorz Waligora[2] formulated the scheduling problem as a linear programming problem and proposed local search metaheuristic to schedule workflow jobs on a Grid. Tracy D. Braun[4] compares eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Ajith Abraham, Rajkumar Buyya[3]hybridized nature's heuristics namely GA, TS and SA and used the hybrid heuristics for job scheduling in a Grid.
Peter Kowling[7]applied hyper heuristics to a real world problem of Personnel Scheduling occurring at a UK academic institution. The hyper heuristics was found to produce results of much higher quality than those obtained by other naïve approaches. Edmund Burke and Yuri Bykov[13]proposed the use of hyper heuristic technique for course time tabling problems. Gonzalez[15]proposed a scheduling method which uses hyperheuristic to schedule independent jobs in computational grid. They considered two modes namely immediate mode and batch mode. The hyperheuristic uses a set of parameters for decision making like the Job heterogeneity, resource heterogeneity and the objective functions makespan, flow time and the resource utilization. Fidanova[16]compared the simulated annealing approach with the ant algorithm for scheduling jobs in Grid.

3 Grid Scheduling

The execution of a job in a dynamic environment like Grid often calls for efficient algorithms to schedule the resources required for successful execution of the jobs. These resources may themselves be dynamic and may enter or leave the system at any point of time or fail and can be idle. So a scheduling strategy is required to generate schedules, which seek to minimize the total execution time of jobs and also adapt to the heterogeneity and the dynamism of the environment. Allocation of a resource to a job, involves three basic steps.

1. Resource Discovery :

   Identifying the resources that are currently free (without being allocated to any job) in the system.

2. Resource Selection :

   Choosing one of the free resources based on some underlying algorithm to schedule it to a job on the ready queue.

3. Job Execution :

   Allocating the chosen resource to a job and executing the job.

The scheduling of Grid jobs can be done using no time characteristics where the resource broker have to make decision based on resource management policies or in the presence of time characteristics derived from the prediction mechanisms. In our work, we consider that the time characteristics are available. The resource providers provide offers based on their local policies and the grid resource broker is responsible for resource discovery, deciding allocation of a job to a particular resource, binding of user applications, initiate computations adapt to changes in grid resources and present the grid to the user as a single unified resource.

3.1 Metaheuristics scheduling

The Grid scheduling problem is a NP complete problem. Various metaheuristics methods are used to solve the scheduling in Grid.

Genetic Algorithm (GA)

GA is the most popular and successful among the metaheuristic that has been used with increasing frequency in the context of scheduling problems. GA is adaptive methods that can be used to solve optimization problems, based on the genetic process of biological organisms. By mimicking this process GA is able to evolve solutions to real world problems if they have been suitably encoded. Much work has been done on using GA for grid scheduling[9, 10, 11, 12] where schedules generated are near optimal. GA uses a direct analogy of natural behavior. They work with a population of individuals each representing a possible solution (partial solution) to a given problem. Each individual is assigned a fitness score according to how good a solution to a problem it is. The highly fit individuals are given opportunities to reproduce by cross-breeding with other individuals in the population. This process is known as mutation and cross-over in technical terms. This produces new individuals as offspring which shares some features of its parents. Thus, by favoring the mating of more fit individuals, the most promising areas of search space are explored. If GA is designed well, the population will converge to an optimal solution of the problem.
The performance of GA depends on the reproduction phase. Here, the chromosomes of the chosen parents are recombined using mechanisms of mutation and crossover. The crossover takes a pair of chromosomes and cuts it at a random position to form two head portions and two tail portions. The tail portions are then swapped to get a new chromosome. Thus the two offsprings get some genes from each of its parent. Mutation is nothing but altering each gene with a randomly chosen small probability value.

Local Search (LS)

LS algorithms use the notion of neighborhood. A neighborhood is a set of solutions possible to generate from the current solution according to a defined neighborhood generation mechanism. These algorithms start from a chosen initial solution and move from one solution to another solution by searching successive neighborhoods to get a solution as close to optimum as possible. The idea of LS (or neighborhood search) has been in existence for a long time and has been used for getting good solutions of the traveling salesman problem. The LS strategy is also used for finding solutions to hard combinatorial optimization problems. One disadvantage LS is that, there are chances of a LS algorithm getting stuck in a Local Maxima or Local Minima.

Simulated Annealing (SA)

SA is a well known metaheuristic that belongs to a class of threshold algorithms. SA exploits the analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure (the annealing process) and the search for a minimum in a more general system. SA works by searching the set of all feasible solutions, and reducing the chance of getting stuck in a poor local optimum by allowing moves to inferior solutions under control of a randomized scheme. Specifically if a move from solution x to a neighboring inferior solution x' results in a change in objective function value by Δc then the move is still accepted if exp(-Δc/T_p)$\geq$Random where T_p (Temperature) is a control parameter and Random $\in$ [0:1] is a uniform random number. The value of T_p is initially high, allowing many inferior moves to be accepted but is gradually brought down to lower the acceptance of inferior moves. Determining the initial value of control parameter and the method of decreasing its value are important considerations in this scheme.

Tabu Search (TS)

TS is a metaheuristic strategy based on neighborhood search with overcoming local optimality. Unlike SA it works in a deterministic way trying to model human memory processes. Memory is implemented by the implicit recording of previously seen solutions, using simple but effective data structures. This approach focuses on the creation of a Tabu list of moves that have been performed recently and are forbidden to be performed for a certain number of iterations, thereby helping to avoid cycling and promoting search in a diversified space. At each iteration, TS moves to the best solution that is not forbidden and thus independent of local optima.

3.2 Hybrid Metaheuristics

To obtain better results, GA is hybridized with other metaheuristics. In our work, GA is hybridized with Local search, Simulated Annealing and Tabu search.

GA-LS

The GA-LS hybrid performs mutations on all the chromosomes, accepting mutated solutions only if they have a better fitness value than the current chromosome that was mutated. This process is repeated for a random number of times, or until a termination condition is met, creating a generation per execution of the process. The population in each generation is sorted according to fitness function and this generation is used as input for the next.

GA-SA

In GA-SA, for every mutation of a given chromosome in the population, a parameter 'temperature' is decreased by a random value. If the mutated solution is better than the current solution, then it is accepted with a probability 1. If the mutated solution is inferior to the current solution, it is accepted with a probability of $e^{-\Delta cost/temperature}$. When the temperature becomes 0(threshold), it is re-initialized to 1. The acceptance of solutions creates a new population. The population in each generation is sorted according to fitness function and this generation is used as input for the next.

GA-TS

GA-TS hybrid maintains a tabu list. Each time a new chromosome is generated by mutating a chromosome from the input population, it is checked to see whether it is present in the tabu list. If it is present, it is rejected, irrespective of its quality with respect to the current solution. If it is not present, the new solution is added to the tabu list and then checked for quality. If it is better than the current solution, it is accepted. This hybrid thus ensures that previously visited solutions are not visited, thus saving running time by avoiding solutions that are already 'seen'.

3.3 Hyper-Heuristics

Metaheuristics are problem-specific solution methods, which require knowledge and experience about the problem domain and properties and require fine-tuning of parameters. Since different metaheuristics have different strengths and weaknesses, it makes sense to see whether they can be combined in some way so that each makes up for the weaknesses of another. A simplistic way of doing this would be as shown below:

```
If (ProblemType(P) = P1) Apply (heuristic1, P)

Else if (ProblemType(P) = P2) Apply(heuristic2,P)

Else . . .
```

One logical extreme of the above approach would be an algorithm containing infinite switch statements enumerating all finite problems and applying the best-known heuristic for each which is certainly not a good approach. A better approach is to associate each heuristic under the problem condition under which each flourishes and hence apply different heuristics to different phases or parts of the solution process.
Hyper-heuristic describes a set of strategies that are used to choose a heuristic from a set of low level heuristics. It can be described as a supervisor, which controls the choice of which local search neighborhood to choose while constructing a solution/schedule. A local search neighbor, also known as a low-level heuristic, is a rule or a simple method that generally yields a small change in the schedule. The strategies used can be very simple or metaheuristics and the hyper-heuristic can also be a metaheuristics. Figure 1 represents the general framework for hyper-heuristic approach.

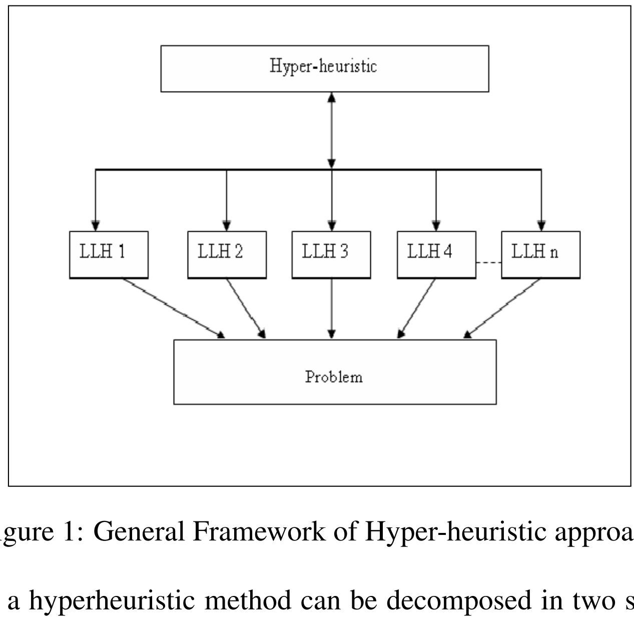

Figure 1: General Framework of Hyper-heuristic approach

A single iteration of a hyperheuristic method can be decomposed in two stages, heuristic selection and movement acceptance. The selection methods can be simple which randomly select the low level heuristics. Greedy hyperheuristics chooses the best performing heuristic at each iteration. Choice function keeps track of previous performance of each heuristic and makes a choice via a choice function. The movement acceptance can be deterministic or nondeterministic. One of the nondeterministic acceptance criteria is the great deluge algorithm. The Hyper-heuristic algorithm used is based on the extended Great

Deluge algorithm. It uses the Greedy selection heuristic to select the best heuristic. The basic concept of the Extended Great Deluge Hyper-heuristic Algorithm is as shown below.
Step 1: Initialization

1. Initialize population.
2. Set the total iterations N;
3. Evaluate fitness function f(s)
4. Initial level B_0 = f(s)
5. Specify input parameter B
6. Set i = 1;

Step 2: Process ith chromosome
While not stopping condition do

1. Select the candidate solution s^* by applying the selection heuristic
2. Evaluate fitness function for s^*
3. If f(s^*) $\leq$ B then accept s^*; f(s) = f(s^*);
4. B = B-Δ B;

During the initialization step, the fitness function f(s_o) is set level B. This is slowly reduced by ΔB at each iteration. The performance of the algorithm is dependent upon the choice of theΔB parameter and is dependent on the number of iterations and initial fitness function.

4 Scheduling Model

Each job to be scheduled for processing has a unique id and an associated processing resource requirement like the number of processing cycles required. The jobs are indivisible, independent of all other jobs. The arrivals of jobs are random and are placed in a queue of unscheduled jobs. Available processing resources vary over time, a exponential smoothing function, A (i) = (1-v) A (i-1) + v a (i-1) is used to smooth out the fluctuations where v is the control parameter chosen between 0 and 1 and the sequential arrival of the processes are given by a(i). The smoothing is done by allowing the recent values to exert more influence than the older values. Batches of jobs from the queue are scheduled on processors during each invocation of scheduler.
At any instant of time, when the number of jobs that have to be scheduled are greater than the number of resources available, then a suitable mapping of jobs to resources have to be found. We follow the following allocation mechanism, in such a case. If R resources and J jobs are available at a time instant(J>R), then a balanced randomized initial population is generated using the most - into - least (MIL) list scheduling heuristics which has been successfully used in GA schedulers. A random number of tasks are assigned to resources in a round robin fashion. The remaining tasks are then sorted and the job requiring the maximum number of cycles (Longest Job) to complete is chosen and is allocated to the resource with the greatest speed (Fastest Resource). The fastest resource is chosen from the list of resources that become available first. This scheduling policy is referred to as the Longest-Job-Fastest-Resource (LJFR) heuristic. Once a job is allocated by the LJFR heuristic, the next job is allocated on similar lines, but using the SJFR (Shortest-Job-Fastest-Resource) heuristic. Thus, the LJFR-SJFR heuristic is applied

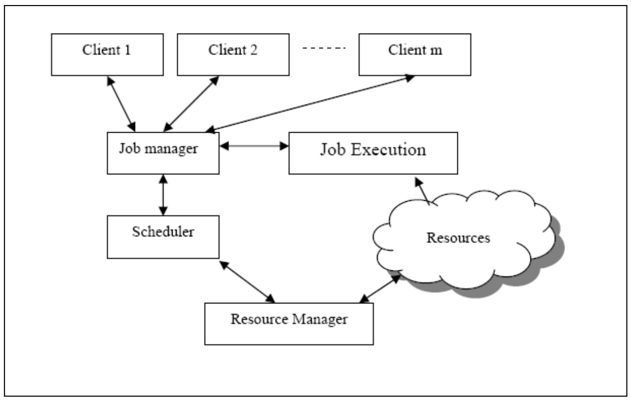

Figure 2: Representation of chromosome

alternatively to the sequence of jobs and resources. This method of allocation is represented by a 'chromosome' as shown in figure 2. Each 'sequence' represents a mini-allocation table, allocating R jobs to R resources. The Expected Time to Compute matrix [4] is formed from the time characteristics of the jobs and stored the ETC matrix of size J×R. The fitness function is based on the makespan of the schedule. It is calculated as min_s max $\{C_i$, i = 1,. . . J$\}$

where C_i is the finishing time of latest job and s is the schedule.

5 Experimental Results

The scheduling experiment was performed for several test cases and the results obtained are graphically represented. For each test case, the experiment was executed at an average of 50 times to measure the performance of the heuristics. The graph in figure 3, 4 and 5 show the comparison of GA-LS, GA-SA, GA-TS and Hyper-heuristic for 50 runs of the sample test cases.

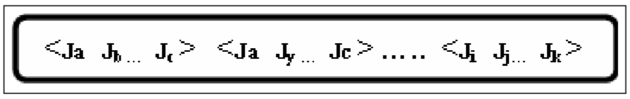
$<J_a\ J_b \ldots J_c> \quad <J_a\ J_y \ldots J_c> \ldots\ldots <J_i\ J_j \ldots J_k>$

Figure 3: Makespan comparison for scheduling 200 jobs on 20 resources

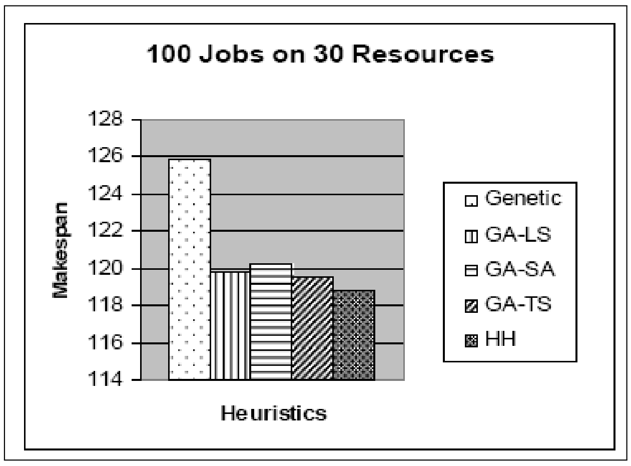

Figure 4: Makespan comparison for scheduling 100 jobs on 30 resources

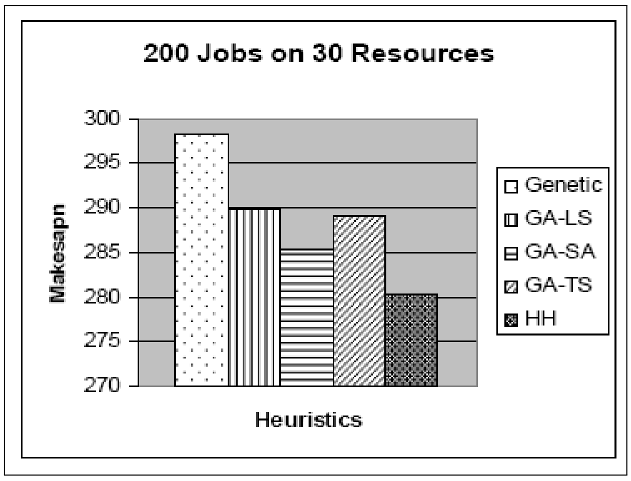

Figure 5: Makespan comparison for scheduling 400 jobs on 250 resources

6 Conclusion

In this paper, we attempted to use hyper heuristics on top of three hybrid Meta heuristics built using four known heuristics, namely GA, TS, SA and LS for scheduling jobs in a grid environment. The hyper heuristic built on top of the hybridized metaheuristics is experimentally shown to give better results than the individual hybrid heuristics in all the test cases.
Global optimization algorithms attract considerable computational effort. In a dynamic environment like Grid, the main emphasis would be to generate schedules in a minimum amount of time. The performance of the scheduler can be improved by using parallel GA. The scheduler in our work is designed for a single objective. Since the resource scheduling problem in grid is a multiobjective one, a Pareto based multiobjective genetic algorithm may be used to generate the schedule sequence.

Bibliography

[1] Moreno, R., Alonso-Conde A.B, Job Scheduling and Resource Management Techniques in Dynamic Grid Environments In et al., F.F.R., ed.: Across Grids 2003, Volume 2970 of *Lecture Notes in computer science*, Springer, pp : 25 - 32, 2004.

[2] Marek Mika, Grzegorz Waligora and Jan Weglarz, A Meta-Heuristic Approach to scheduling Workflow jobs on a Grid, *Grid resource management: state of the art and future trends*, ISBN : 1-4020-7575-8 , Kluwer Academic Publishers, Norwell, MA, USA; pp : 295 - 318, 2004.

[3] Ajith Abraham, Rajkumar Buyya and Baikunth Nath, Nature's Heuristics for Scheduling jobs on Computational Grids, *International Conference on Advanced Computing and Communications 2000*, 2000.

[4] Tracy D.Braun, Howard Jay Siegel and Noah Beck, A Comparison of eleven static heuristics for mapping a class of tasks to heterogeneous Distributed Computing System, *Journal of Parallel and Distributed Computing* Vol:61, pp: 810 - 837, 2000.

[5] Edmund Burke, Yuri Bykov, James Newall and Sanja Petrovic, A Time-Predefined Approach to Course Timetabling, *Yugoslav Journal of Operations Research*, 13(2), pp : 139-151, 2000.

[6] Jarek Nabrzyski, Jennifer M. Schopf and Jan Weglarz, *Grid Resource Management - State of Art and Future Trends*, ISBN : 1-4020-7575-8 , Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[7] Peter Kowling, Graham Kendall and Eric Soubeiga, Hyper - heuristics : A tool for rapid prototyping in scheduling and optimization, *LNCS 2279, Applications of Evolutionary Computing : Proceedings of EvoCop2002*, Kinsale, Ireland, 2002.

[8] Ian Foster, Carl Kesselman and Steven Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations, *International J. Supercomputer Applications* 15(3), 2001.

[9] Mona Aggarwal, Robert D. Kent and Alionne Ngom, Genetic algorithm based scheduler for Computational Grids, *International Symposium on High performance Computing Systems and Applications(HPCS'05)*, IEEE, 2005.

[10] Vincenzo Di Martino, SubOptimal Scheduling in a Grid using Genetic Algorithms, *Parallel and nature-inspired computational paradigms and applications* , Elsevier Science Publishers pp: 553 - 565, 2004.

[11] Javier Carretero and Fatos Xhafa, Use of Genetic Algorithms for Scheduling Jobs in Large scale Grid applications, Okio Technologies IR Ekoonominis Vvstymas, *Technological and Economic development of Economy*, Vol XII, No.1 pp 11-17.

[12] Andrew J. Page and Thomas J. Naugton, Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing, *IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, 2005.

[13] Burke, E K. Kendall, G., Newall, J., Hart E., Ross,P., Schulnenburg, *Handbook of Metaheuristics*, Chapter 16, Hyper-heuristics: an emerging direction in modern search technology, pp 457-474, Kluwer Academic Publishers, 2003.

[14] Lingyun Yang, Jennifer M. Schopf and Ian Foster, Conservative Scheduling: Using predictive variance to improve scheduling decisions in Dynamic Environments, *SuperComputing 2003*, November 15-21, Phoenix, AZ, USA.

[15] Juan Antonio Gonzalez, Maria Serna and Fatos Xhafa,2007, A Hyper-heuristic for scheduling independent jobs in Computational Grids, *International conference on software and data technologies, ICSOFT (2007).*

[16] Stefka Fidanova, Simulated annealing for Grid Scheduling problem,*International Symposium on Modern Computing (JVA'06)* IEEE, 2006.

S. Mary Saira Bhanu
Department of Computer Science and Engineering
National Institute of Technology
Tiruchirappalli, India
E-mail: msb@nitt.edu

N.P.Gopalan
Department of Computer Applications
National Institute of Technology
Tiruchirappalli, India
E-mail: gopalan@nitt.edu

Received: December 20, 2007

S.Mary Saira Bhanu received the B.E Degree in Electronics and communication from Madurai Kamaraj University in 1986 and the M.E Degree in Computer Science from Bharathidasan University in 1989. Currently, she is a Lecturer in the Department of Computer Science and Engineering in National Institute of Technology, Tiruchirappalli, India. Her research interests include Distributed, Parallel and Grid Computing.

N.P. Gopalan received the M.Sc. from Madras University in 1978 and the PhD in applied mathematics from Indian Institute of Science, Bangalore, in 1983. Currently, he is a professor in the Department of Computer Applications in National Institute of Technology, Tiruchirappalli, India. His research interests include algorithms, combinatorics, data mining, and distributed, parallel and grid computing.