Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 6, June 2015, pg.611 – 619

SURVEY ARTICLE

A Survey of Evolutionary Heuristic Algorithm for Job Scheduling in Grid Computing

Kanwerjit Singh[1]
GNDU, Amritsar
Kanwerjit91@gmail.com

Amit Chhabra[2]
(Asst. professor)
Dept. of computer Engg. and Technology
GNDU, Amritsar

Abstract- An efficient management of the resources in Grid computing crucially depends upon the efficient mapping of the jobs to resources according to the user's requirements. Grid resources scheduling has become a challenge in the computational Grid. The mapping of the jobs to appropriate resources for execution of the application in Grid computing is an NP-Complete problem. So there is no best solution for all grid computing system. Job and resource scheduling in grid environment is one of the key research area in grid environment. The comparison of the heuristic has been shown and experimental result shows that the hyper-heuristics can be of significance importance in Grid scheduling. Over the time, heuristics and meta-heuristics have proved to provide an optimum solution for the combinatorial optimization problems. In this paper, a survey of scheduling algorithms and heuristic approaches is done.

Keywords- Resource scheduling, Grid computing, Heuristic approach, Hyper Heuristic approach

1. Introduction

Grid computing system have become popular for the resolution of large-scale complex problem from science, engineering, finance etc. The ultimate goal of Grid computing is to provide the computing facility to users like power Grid without knowing the detailed characteristics of the source. So Grid resource management has become one of the most important areas of Grid computing. Grid resource management can be defined as a process of identifying requirements of the resources, matching resources to the applications, allocating those resources and finally scheduling and monitoring the Grid resources over time in order to run Grid applications as efficiently as possible. Grid resource management system is required to perform resource management decisions which include resource provisioning and scheduling, while maximizing the Quality of Service (QoS) metrics delivered to the clients [1]. The base of Grid computing is a resource. To manage resources in Grid environment is a challenging task.

In order to accomplish a job requiring large-scale computation over a distributed system, it is difficult to have a fair judgment on which dispatching method leads to optimal solution. However, mapping independent task on to a

heterogeneous computing (HC) suite is a well-known NP Complete Problem [2]. NP-Complete problems are often solved using heuristic methods.
Heuristic approaches can be easily applied to Grid scheduling problems because Grid scheduling has various important issues such as heterogeneity of resources, dynamic and autonomous nature of Grid resources and finally there source providers and resource consumers have different policies for execution of their applications. Hyper-heuristic can be seen as a high-level methodology, which when given a particular problem instance or a class of instances and a number of low-level heuristics, automatically produced an adequate combination of the provided components to effectively solve the given problems [3]. We analyze why heuristic and meta heuristic methods are good alternatives to more traditional scheduling techniques and what make them appropriate for grid scheduling [5].
The paper is structured as follows: Section 2 discusses related work. In section 3, we discussed some scheduling problems in grid systems and phases of scheduling in grid computing as well. In section 4, we discussed different heuristic and meta heuristic approaches for scheduling in grid computing. Section 5, we presents the Analysis and comparison between various heuristic approaches.

2. Related work

Abraham et al. used nature's heuristics namely Genetic Algorithm (GA), Simulated Annealing (SA) and Tabu Search (TS) for scheduling of jobs on computational Grids. Authors illustrated that GA performs better than TS and SA for scheduling of the jobs to exact resources but hybrid heuristic algorithms perform better than GA approach as it minimizes the time required for scheduling the job [5]. Aron et al. defines the hyper heuristic based resource scheduling approaches in grid computing. Authors proposed a novel hyper-heuristic based scheduling algorithm for scheduling of jobs in Grid environment so as to minimize the cost and time by minimizing the makespan [4]. Xhafa et al. illustrated the scheduling problem in grid system and different phases of scheduling in grid environment[6]. Channa et al. discussed all the heuristic based scheduling approaches for grid scheduling [10].

3. Scheduling problems in Grid systems

The scheduling problem is one of the most studied problems in the optimization research community. However, in the Grid setting there are several characteristics that make the problem different and more challenging. Some of these problems are the following[6].
The dynamic structure of the Computational Grid: Unlike traditional distributed systems, resources in a Grid system can join or leave the Grid in an unpredictable way. This could be simply due to losing connection to the system or because their owners switch off the machine or change the operating system, etc. Given that the resources cross different administrative domains, there is no control over the resources.
The high heterogeneity of resources: In Grid systems, computational resources could be very disparate in their computing capacity, ranging from laptops, desktops, clusters, supercomputers and even small computational devices. Current Grid infrastructures are not yet much versatile but heterogeneity is among most important features in any Grid system.
The high heterogeneity of jobs: Jobs arriving at any Grid system are diverse and heterogeneous in terms of their computational needs. For instance, they could be computing intensive or data intensive; some jobs could be full applications having many specifications and others could be just atomic tasks. Importantly, in general the Grid system will not be aware of the type of tasks arriving in the system.
The high heterogeneity of interconnection networks: Grid resources are connected through the Internet using different interconnection networks. Transmission costs will often be very important in the overall Grid performance and hence smart ways to cope with the heterogeneity of interconnection networks is necessary.
The large scale of the Grid system: Grid systems are expected to be large scale. Similarly, the number of jobs, tasks or applications submitted to the Grid over time could be large as well. Therefore, the efficient management of resources and planning of jobs will require the use of different types of scheduling (super-schedulers, meta-schedulers, decentralized schedulers, local schedulers, resource brokers, etc.) and their possible hierarchical combinations to achieve scalability.

The existence of local policies on resources: Again, due to the different ownership of the resources, one cannot assume full control over the Grid resources. Companies might have unexpected computational needs and may decide to reduce their contribution to the Grid. Other policies on access, available storage, pay-per-use, etc. are also to be taken into account.

The job-resource requirements: Current Grid schedulers assume full availability and compatibility of resources when scheduling. In real situations, however, many restrictions and/or incompatibilities could be derived from job and resource specifications.

Security: This characteristic, which is non-existent in classical scheduling, is an important issue in Grid scheduling. Here the security can be seen as a two-fold objective: on the one hand, a task, job or application could have security requirements and, on the other hand, the Grid nodes could have their own security requirements.

3.2 Phases of scheduling in Grids

In order to perform the scheduling process, the Grid scheduler has to follow a series of steps which could be classified into five blocks[5]: (1) Preparation and information gathering on tasks submitted to the Grid; (2) Resource selection; (3) Computation of the planning of tasks to selected resources; (4) Task (job or application) allocation according to the planning (the mapping of tasks to selected resources); and (5) Monitoring of task completion.

Preparation and information gathering: Grid schedulers have access to the information on available resources and tasks through the Grid Information Service. Moreover, the scheduler will be informed about updated information (according to the scheduling mode) on jobs and resources.

Resource selection: Not all resources could be candidates for the allocation of tasks. Therefore, the selection process is carried out based on job requirements and resource characteristics. The selection process, again, will depend on the scheduling mode. For instance, if tasks were to be allocated in a batch mode, a pool of as many as possible candidate resources will be identified out of the set of all available resources. The selected resources are then used to compute the mapping that meets the optimization criteria.

Computation of the planning of tasks: In this phase the planning is computed.

Task allocation: In this phase the planning is made effective: tasks are allocated to the selected resources according to the planning.

Task execution monitoring: Once the allocation is done, the monitoring will inform about the execution progress as well as possible failures of jobs, which depending on the scheduling policy will be rescheduled or migrated to other resources.

4. Heuristic and meta-heuristic approaches for scheduling in Grids

From the exposition in the previous sections, it is clear that the Grid scheduling problem is really challenging. Dealing with the many constraints and optimization criteria in a dynamic environment is very complex and computationally hard. Meta-heuristic approaches are undoubtedly considered the de facto approach. We now point out the main reasons that explain the strength of meta-heuristic approaches for designing efficient Grid schedulers.

Meta-heuristics are well understood: Meta-heuristics have been studied for a large number of optimization problems, from theoretical, practical and experimental perspectives. Certainly, the known studies, results and experiences with meta-heuristic approaches are a good starting point for designing meta heuristic- based Grid schedulers.

No need for optimal solutions: In the Grid scheduling problem, for most practical applications good quality planning of jobs would suffice rather than searching for optimality. In fact, in the highly dynamic Grid environment, it is not possible to even define the optimality of planning, as it is defined in combinatorial optimization. This is so due to the fact that Grid schedulers run as long as the Grid system exists and thus the performance is measured not only for particular applications but also in the long run. It is well known that meta-heuristics are able to compute in a short time high-quality feasible solutions. Therefore, in such situation meta-heuristics are among best candidates to cope with Grid scheduling.

Efficient solutions in short time: Research work on meta heuristics has by large tried to find ways to avoid getting stuck in local optima and ensure convergence to suboptimal or optimal solutions. that allow one to tune the convergence speed.
For instance, in Genetic Algorithms, by choosing appropriate genetic operators one can achieve a very fast convergence of the algorithm to local optima. Similarly, in the Tabu Search method, one can work with just short-term memory in combination with an intensification procedure to produce high-quality feasible solutions in a very short time. This feature of meta-heuristics is very useful for Grid schedulers in which we might want to have a very fast reduction in makespan, flow time and other parameters.
Dealing with multi-objective nature: Meta-heuristics have proven to efficiently solve not only single-objective optimization problems but also multi-objective optimization problems.
Appropriateness for periodic and batch scheduling: Periodic scheduling is a particular case of Grid scheduling. It arises often when companies and users submit their applications to the Grid System periodically. In this case resource provisioning can be done in the Grid infrastructures and, which is more important in our context, there are no strong time restrictions. This means that we can run meta-heuristic-based schedulers for longer execution times and significantly increase the quality of planning of jobs to resources. Similarly, in batch scheduling, we could run the meta-heuristic-based scheduler for the time interval comprised within two successive batches activations.
Hybridization with other approaches: Meta-heuristics can be easily hybridized with other approaches. This is useful to make Grid schedulers to better respond to concrete Grid types, specific types of applications, etc. The hybridization in general can produce better solutions than those delivered by single approaches.
Designing robust Grid schedulers: The changeability of the Grid environment over time is among the factors that directly influences the performance of the Grid scheduler. A robust scheduler should deliver high-quality planning even under frequent changes of the characteristics of the Grid infrastructure. Evidence in meta-heuristics literature exists that in general meta-heuristics are robust approaches.
In the following subsections we briefly review the most important heuristic approaches and the benefits of using them for the Grid scheduling[10].

4.1. Local search-based heuristic approaches

Local search heuristic approaches is a family of methods that explore the solution space by starting at an initial solution, and constructs a path in solution space during the search process. Local search heuristic approaches improve solutions through neighborhood search. The main objective of this local search based heuristic approach is to gain feasibility as soon as possible. They have been applied successfully to many industrial problems and performance of local search based heuristic approaches depending on construction of neighborhood.

Tabu Search Tabu Search (TS) is a high level heuristic procedure for solving optimization problems and was proposed by Glover in 1986. Tabu search is a meta-heuristic that guides a local search procedure to explore the solution space beyond local optimality [5].
Advantages
TS avoids entrapment in local minima and continues the search to give a near optimal final solution.
TS is very general and conceptually much simpler than other meta heuristic algorithms such as genetic algorithm, simulated annealing and ant colony optimization algorithms.
TS is very easy to implement.
TS does not require special memory space.
TS takes short searching time to solve combinatorial optimization problems.
TS uses specific set of constraints, known as tabu conditions, in order to avoid blind search.

Disadvantages
TS often gets locked in looping from one local optimum to another.
TS has low global search capability.

Hill Climbing: Hill Climbing (HC) is a graph search algorithm where the current path is extended with a successor node which is closer to the solution than the end of the current path. Hill climbing is logical and beneficial especially in situations where the search space is of simple nature with no more than a single maxima or minima.

Advantages

- HC is a local search heuristic technique.
- Hill climbing is simpler and straight forward in comparison to other heuristics.

Disadvantages

- In case of hill climbing, the solution is better than all of its neighbors, but it is not better than some other states far away.
- There is a flat area of the search space in which all the neighboring states have the same value.
- HC is a local method and it moves in many directions at a time.

Simulated Annealing: Simulated Annealing (SA) heuristic approach was proposed by kirkpatrick et al in 1983. The simulated annealing process consists of first melting the system being optimized at high effective temperature, then lowering the temperature by slow stages untill the system freezes and no further changes occur [7]. SA is an iterative technique that considers only one possible solution (mapping) for each meta task at a time [8]. This solution uses the same representation as the chromosome for the genetic algorithm. The initial implementation of SA was evaluated and then it was modified and refined to give a better final version. SA uses a procedure that probabilistically allows poorer solutions to be accepted in an attempt to obtain a better search of the solution space.

Advantages

- SA is guaranteed to converge in asymptotic time.
- SA can deal with arbitrary systems and cost functions.
- SA statically guarantees to find an optimal solution.
- SA is relatively easy to code, even for complex problems.
- SA is a robust heuristic to implement and has an ability to provide reasonably good solutions for many combinatorial problems.

Disadvantages

- SA has a difficulty in defining a good cooling schedule which is important both in single and multi-objective optimization.
- In case of SA, if there is a repeated annealing then the scheduling is very slow, especially if the cost function is expensive to compute.
- SA is often comparable to heuristics.
- The main drawback of simulated annealing is that there is a need for a great deal of computer time for many runs and carefully chosen turnable parameters.

4.2 Population-based heuristic approaches

Population-based heuristics is a large family of methods that have shown their efficiency for solving combinatorial optimization problems. Population-based methods usually require large running times if suboptimal or optimal solutions are to be found. However, when the objective is to find feasible solutions of good quality in short execution times, as in the case of Grid scheduling, we can exploit the inherent mechanisms of these methods to increase the convergence of the method.

Genetic Algorithms Genetic Algorithm (GA) was proposed by holland et al. Genetic algorithms are playing an increasingly important role in studies of complex adaptive systems, ranging from adaptive agents in economic theory to the use of machine learning techniques in the design of complex devices such as aircraft turbines and integrated circuits [14]. GA is a famous stochastic optimization algorithm which uses biologically inspired techniques such as genetic inheritance, natural selection, mutation and sexual reproduction (recombination, or crossover). Genetic algorithms are useful heuristics to find a near optimal solution in large search spaces [15]. In GA, a point in search space is represented by a set of parameters and these parameters are known as genes and a set of genes is known as string or a chromosome. A fitness function must be devised for each problem to be solved. Each chromosome is assigned a fitness value that indicates how closely it satisfies the desired objective. Given a particular chromosome, the fitness function returns a single numerical fitness or figure of merit, which will determine the ability of the individual, which that chromosome represents. A set of chromosomes is called population. Reproduction is another critical attribute of GAs where two individuals selected from the population are allowed to mate to produce offspring, which will comprise the next generation. Having selected two parents, their chromosomes are recombined, typically using the mechanisms of crossover and mutation. Mutation provides a small amount of random search, and helps ensure that no point in the search space has a zero probability of being examined. If the GA has been correctly implemented, the population will evolve over successive generations so that the fitness of the best and the average individual in each generation increases towards the global optimum. The genetic algorithms have been found to be very powerful in finding out a global minima.

Advantages

- In case of GA, there is no need of analytical knowledge.
- GA is easy to understand and implement.
- GA supports multi objective optimization.
- GA is easy to parallelize and no derivatives are required.
- GA works on a wide range of problems and has better global capability.

Disadvantages

- Genetic algorithm requires much more evolution functions than linearized methods.
- There is no guaranty of convergence to a local minima.
- It converges to a local optima or an arbitrary point rather than the global optima of the problem.
- GA has a slow convergence rate and premature convergence.
- It cannot use the feedback of a system.

Memetic Algorithm Memetic Algorithm (MA) is an extension of genetic algorithm. Memetic algorithms are evolutionary algorithms that can be applied on a local search process to refine solutions for hard problems. Memetic algorithms are the subject of intense scientific research and have been successfully applied to a multitude of real-world problems ranging from the construction of optimal university exam timetables, to the prediction of protein structures and the optimal design of space-craft trajectories.

Advantages

- Memetic algorithm can handle complex objective functions.
- It combines the advantages of local search and genetic algorithm for optimization problems.
- MA can be used for global search.
- It is based on a genetic algorithm and extended by a search technique to further improve in dividual fitness that may keep with the population diversity and reduce the likelihood premature convergence.

Disadvantages

- MA requires a considerable amount of time and memory needed for improvement of its performance.
- MA can be used only in non-linear continuous multi objective combinatorial optimization problems.

Ant Colony Optimization Ant Colony Optimization (ACO) was proposed by Marco Dorigo in 1992 [11].The real power of ants resides in their colony brain. The self-organization of those individuals is very similar to the organization found in brain-like structures. Like neurons, ants use mainly chemical agents to communicate. One ant releases a molecule of pheromone that will influence the behavior of other ants. Ant algorithms are often compared with other evolutionary approaches such as Genetic Algorithms, Evolutionary Programming and Simulated Annealing. It is important to remember that Ant algorithms are non-deterministic and rely on heuristics to approximate to a sub-optimal solution in cases where the number of combinations is extremely huge and is impossible to calculate using a deterministic algorithm

Advantages

- ACO is versatile and can be applied to similar versions of the same problem; for example, there is a straightforward extension from the Traveling Salesman Problem (TSP) to the Asymmetric Traveling Salesman Problem (ATSP).
- It is robust and can be applied with only minimal changes to other combinatorial optimization problems such as the Quadratic Assignment Problem (QAP) and the Job-Shop Scheduling Problem (JSP).
- It is a population based approach. This is interesting because it allows the exploitation of positive feedback as a search mechanism. It also makes the system amenable to parallel implementations.
- ACO convergence is guaranteed and it can be used for solving constrained discrete problems.
- ACO has the powerful feedback capability which can increase the speed of evolution of algorithm to make algorithm convergence possible in the end.

Disadvantages

- ACO's convergence rate is slow in comparison to other heuristics.
- ACO performs poorly for larger city in Traveling salesman problems.
- In ACO, there is no centralized control to guide and provide good solutions.
- ACO can be applicable to only discrete problems and theoretical analysis in ACO is difficult.

Particle Swarm Optimization Particle Swarm Optimization (PSO) is a method for performing numerical optimization without explicit knowledge of the gradient of the problem to be optimized. PSO is one of the latest evolutionary optimization techniques inspired by nature and was introduced in 1995 by Kennedy and Elberhart [11]. It simulates the process of a swarm of birds preying. It has the better ability of global searching and has been successfully applied to many areas. A flock or swarm of particles is randomly generated. Initially, each particle position represents a possible solution point in the problem space. The fitness value of each particle is evaluated by the objective function to be optimized. Each particle remembers the coordinates of the best solution (gbest) achieved so far. The coordinates of current global best (pbest) are also stored.

Advantages

- PSO is a robust stochastic optimization based on the movement and intelligence of swarms.
- There is no selection and crossover parameter like genetic algorithm
- PSO is easy to implement, few parameters to adjust, computationally efficient etc.
- PSO is efficient for global search algorithm.

Disadvantages

- PSO has a weak local search.
- PSO has a slow convergence rate in refined search strategy.

4.3 Hybrid heuristic approaches

Hybrid strategies have been constructed to exploit the meta heuristic techniques. To get a better result of genetic algorithm, it has been hybridized with local search methods as tabu search and simulated annealing etc. The major advantage of parallel hybrids implemented on shared-memory parallel architectures is their simplicity.

For instance, MAs combine an evolutionary search with a local search. However, hybridization among different meta-heuristics has been shown to be effective for many problems by outperforming single methods. However, hybrid meta-heuristics have been less explored for the problem. Abraham et al. [7] addressed the hybridization of GA, SA and TS heuristics; GA+SA hybridization is expected to have a better convergence than a pure GA search and GA+TS could improve the efficiency of the GA. In these hybridizations a heuristic capable to deal with a population of solutions, such as a GA, is combined with local search heuristics, such as TS and SA, that deal with only one solution at a time.

Advantages

- Hybrid-heuristics have a better convergence.
- hybrid-heuristics are more efficient in comparison to genetic algorithm.

Disadvantages

- Hybrid-heuristics are not easy to implement.
- Hybrid-heuristics are time consuming.

4.4 Hyper-heuristic approaches

Hyper-heuristics are methods that guide the search, at a higher level as compared to meta-heuristics. Hyper-heuristics have proven to be effective for scheduling and timetabling (Burke et al. [3]). The Hyper heuristic is a high level algorithm. A hyper-heuristic can be seen as a high level methodology which when given a particular problem instance or a class of instances and a number of low-level heuristics, automatically produces an adequate combination of the provided components to effectively solve the given problems.

Advantages

- Hyper-heuristics operate in a space of heuristics, choosing and applying one low-level heuristic from a given set at each decision point.
- Hyper-heuristics do not require knowledge of each low level heuristic.
- Hyper-heuristics are robustness and re-applicability heuristics.

Disadvantages

- For many hyper-heuristics, a significant amount of parameter tuning is required in order to find good parameter settings for a given problem.
- A large number of problem instances may be required for training and testing of the method in order to accumulate enough knowledge to make the right choice of low-level heuristics.

Analysis and Comparison between various heuristic approaches

Considering all these criteria it is found that hyper-heuristic provides a better solution and near optimal solution with low cost per second and minimum make span for Grid scheduling problems. It can be concluded that Grid scheduling is one of the main challenging issues of Grid computing. Meta-heuristics are highly adaptive in Grid computing environment but it does not provide good solutions for more number of jobs in heterogeneous environment.

Heuristic Approaches	Makespan	Cost	Parameters	Services	Local/Global Search	Optimization Problems
Tabu Search	High	High	Less Parameters	Simpler, easy to implement, no special memory required	Local search	Combinatorial Optimization
Hill Climbing	High	High	Less	Simpler, staright forward	Local search	Simple optimization
Simulated Annealing	High	Moderate	Less	Easy to code, robust heuristic	Local search	Combinatorial Optimization
Genetic Algorithm	Moderate	High	More Function	Easy to implement	Global Search	Multi objective optimization
Memtic Algorithm	Moderate	Moderate	More function	Flexible	Global search	Complex objective function, combinatorial
Ant Colony Optimization	Moderate	High	Less function	Versatile, robust	Global	Static and Dynamic optimization
Hybrid Heuristic	Low	Low	More functions	Flexible	Global	Stochastic optimization
Hyper-Heuristic	Low	Low	More function	Flexible, robust No. of meta heuristic required	Global	Real world optimization, complex

Table 1: Comparison of different heuristic approaches

5. Further Issues

Besides the many aspects and facets of the Grid scheduling problem presented in the previous sections, there still remain other issues to be considered. We briefly mention some of them here.

In this survey we consider hyper heuristic is near optimal solution but there is still chances of improvement by using other techniques like Bacterial Foraging Optimization.

Security is an another important aspect to be considered in Grid scheduling. It is challenging to incorporate the security/trust level as one of the objectives of the scheduling.. Moreover, the objective is to reduce the possible overhead to the Grid scheduler that would introduce a secure scheduling approach.

Most current Grid approaches are task-oriented or resource-oriented approaches. For instance, tasks are assumed to include all data needed for its computation or tasks are just the processes and data is assumed to be available in Grid nodes. However, with the ever-increasing complexity of large-scale problems in which both tasks and data are to be scheduled, an integrated scheduling approach that would optimize allocation of both the task and the data is required.

References

1. Foster, I. and Kesselman, C., ”The Grid: Blueprint for a Future Computing Infrastructure”, Morgan Kaufmann Publishers, USA, 2004.

2. Lee, A. and Parashar, M., Senior member of IEEE,” A Survey of Job Scheduling and Resource Management in Grid Computing”, 2008.

3. Buyya, R. and Venugopal, S.,”A Gentle Introduction to Grid Computing and Technologies” Computer Society of India, july 2005.

4. Bhuyan, P., Sharma, R., Soni, V.K. and Mishra, M.K.,” A Survey of Job Scheduling and Resource Management in Grid Computing”, World Academy of Science and Engg And Tech,2010.

5. Burke,E.K., Hyde,M., Kendall, G., Ochoa, G., Ozcan,E. and Qu, R., ”Hyper-heuristics: A survey of the State of the Art”,Technical report, University of Nottingham, 2009.

6. Xhafa, F. and Abraham, A., Computational models and heuristic methods for Grid scheduling problems” Future Generation Computer System 2010.

7. Abraham,A., Buyya,R. and Nath,B., ”Nature’s Heuristics for Scheduling Jobs on Computational Grids”. The 8th IEEE Conference on Advanced Computing and Communications, Cochin, India, 2000.

8. Abraham, I., Aron, R. and Chnna, I.," Hyper-Heuristic Based Resource Scheduling in Grid Environment”, IEEE International Conference on Systems, 2013.

9. Aron, R. and Channa, I.,” Bacterial foraging based hyper-heuristic for resource scheduling in Grid computing”, Department of Computer Science and Engineering, Thapar University, Patiala, India, September 2012.

10. Aron, R. and Chnna, I., ”Grid scheduling heuristic methods: State of the Art”, ISSN 2150-7988 Volume6 (2014).

11. Liu, H., Abraham, A. and Hassanien, A.E., “Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm”, Future Generation Computer Systems, 2010.

12. Pooranian, Z., Harounabadi, A. and Hedayat, N., “New hybrid Algorithm For Task Scheduling in Grid Computing to Decrease missed Task ”, world academy of Science, Engg and Tech,2011.

13. Passino, K.M.,”Biomimicry of Bacterial Foraging for Distributed Optimization and Control”,IEEE Control and System Magazine, 2002.

14. Garg, S., Konugurthi, P.and Buyya, R.,” A linear programming driven genetic algorithm for meta scheduling on utility Grids”, in:16th International Conference on Advanced Computing and Communication, ADCOM 2008, IEEE Press, New York, USA, 2008.