Adaptive Task Scheduling in Grid Computing Environments

Angelos Michalas
Department of Informatics and Computer Technology
Technological Educational Institute of Western Macedonia
Kastoria 52100, Greece
amichalas@kastoria.teikoz.gr

Malamati Louta
Department of Informatics and Telematics
Harokopio University of Athens
Athens 17671, Greece
louta@hua.gr

Abstract— **In this study the task scheduling problem in Grid computing environments has been addressed. To resolve the problem a set of Grid Services are defined and implemented conforming to the OGSA standards. The proposed scheduling architecture is semantically enhanced and provides the most appropriate assignment of tasks to computing resources, given the current load conditions of each computing resource and the network status. The Ant Colony Optimization algorithm (ACO) was used to effectively assign tasks to computing resources. First experimental results indicate that the proposed framework produces good results in comparison to other well known schemes considered (e.g. round robin).**

Keywords- grid services; semantics OWL; task scheduling; load balancing; ant colony algorithm

I. INTRODUCTION

Powerful computers availability and high-speed network technologies penetration are changing the way computers are used. These technology enhancements led to the possibility of using distributed computers as a single, unified computing resource, leading to what is popularly known as Grid computing [1].

The term Grid is adopted from the power Grid which supplies transparent access to electric power regardless of its source. Cloud computing, scalable computing, global computing, internet computing, and more recently peer-to-peer computing are well known names describing the Grid technology in distributed systems.

Grids facilitate the employment of various resources comprising supercomputers, storage elements and databases that are distributed for resolving computational demanding problems in many disciplines of science and commerce [1].

Service Oriented Architecture (SOA), Web Services and Grid computing are converging, leading to the Open Grid Services Architecture (OGSA) [2]. The advances in Web Services technology acted as a catalyst for the evolution from the primitive architecture of current Grids to the sophisticated service-oriented Grids. OGSA [3] has enhanced Web Services properties by defining the concept of Grid Services as having architectural features such as: service factories, registries, naming and referencing standards for service instances, stateful services, event notification mechanisms and versioning support.

To utilize Grids effectively, an efficient allocation algorithm is needed to assign tasks to Grid resources. The aim of this paper is to contribute to the intelligence of Grid Services Architecture by proposing an adaptive scheduling mechanism based on the Ant Colony Optimization (ACO) algorithm whose foundations lie in the nature. Specifically, assuming that a user wishes to perform a specific task which can be served by various candidate computing resources (CCRs), a problem that should be addressed is the assignment of the requested task to the most appropriate computing resource. In this paper, the pertinent problem is called Adaptive Task Scheduling (ATS). The paper considers the problem by defining a Grid service architecture in which services are used to vitalize resources and grid management functions.

This study is related to the pertinent previous work in the literature, since efficient resource utilization, load balancing and job scheduling are topics that attract the attention of the researchers, as computational Grids have become an emerging trend on high performance computing. Most studies in the field of resource allocation schemes aim at efficiently utilizing the otherwise unutilized computing power spread throughout a network. Different global objectives could be considered, such as minimization of mean task completion time, maximization of resources utilization (e.g., CPU time), and minimization of mean response ratio, while in most cases load balancing among resources is considered. ACO algorithm exploited in the current version of this study, is used to solve many NP-hard problems including routing, assignment, and scheduling. ACO actions follow the behavioural pattern of real ants in nature, which travel across various paths marking them with pheromone while seeking for food.

A high level problem statement addressed in the current version of this study may be the following. Given the set of candidate computing resources and their layout, the set of application tasks to be executed, the resource requirement of each task in terms of CPU utilization, the characteristics of each computing resource, the current load conditions of each computing resource and of the network links, find the best assignment pattern of tasks to computing resources subject to a set of constraints, associated with the capabilities of the computing resources.

The contribution of this study is a) to propose an intelligent mechanism for adaptive task scheduling to the

most appropriate resource taking into account the current conditions and b) to present a Grid Services Architecture (GSA) supporting the aforementioned mechanism. To this respect a set of grid services are defined conforming to the OGSA standards, semantically enhanced so as to provide a common semantic understanding between the components involved in the scheduling process. The proposed ATS scheme handles complex services composed by tasks requiring communication (i.e., message exchange) with other services (e.g., databases). Care is also taken in case there is no resource with available spare capacity so as to accommodate a new task on a congested system.

The rest of this paper is structured as follows. Section II briefly revisits the related research literature. In Section III the Scheduling Architecture is presented, while the adaptive mechanism for task allocation is described in detail in Section IV. In Section V indicative results are provided in order to assess the proposed mechanism. Finally Section VI concludes the paper and highlights some issues to be considered in the future.

II. Related Work

Most studies in the field of resource allocation schemes aim at efficiently utillising the resources spread throughout a network. In most cases the problem is reduced to load balancing among specific nodes. Basic task scheduling strategies comprise the following [4]: First, Round Robin, according to which the tasks are allocated to nodes by simply iterating through the nodes list. Second, Random, where the nodes to be assigned with the tasks are selected randomly. Third, Least Loaded in accordance with which the tasks are assigned to a specific node until a pre-specified threshold is reached. Thereafter, all subsequent requests are transferred to the node with the lowest load and the aforementioned steps are repeated. Fourth, Load Minimum, where the average load of the system is calculated. In case the load of a node is higher than the average node and of the least loaded node by a certain amount, all subsequent requests are transferred to the least loaded location.

Researchers also borrow notions from economic fields (particularly, dynamic pricing and game theory) in order to efficiently allocate network resources through the construction of market-based systems [5]. In [6], a computational economy framework for resource allocation and for regulating supply and demand in grid computing environments is proposed. Specifically, economic models (commodity market models, posted pricing schemes, tender and auction mechanisms), system architectures and policies for resource management are provided for computational grids and peer-to-peer computing systems.

Currently researchers use service oriented models for Grid resource allocation and management. [7] addresses the problem of finding on a SOA environment the set of service providers that minimizes the total execution time of business process subject to cost and execution time constraints.

In [3] a service oriented Middleware Framework for Grid computing is presented which supports interaction services, resource discovery, resource management and Grid security. In [8] two models for predicting the completion time of jobs in a service Grid are proposed. The single service model predicts the completion time of a job in a Grid that provides only one type of service. The multiple services model predicts the completion time of a job that runs in a Grid which offers multiple types of services.

ACO algorithms are based in a behavioral pattern exhibited by ants and more specifically their ability to find shortest paths using pheromone, a chemical substance that ants can deposit and smell across paths. These algorithms have been emerged in the early ‘90s for the solution of optimization problems. One of the problems ACO tries to solve is the Generalized Assignment Problem (GAP), where a set of tasks has to be assigned to a set of resources. The first ACO application to the GAP was presented by [9] and is called Max-Min Ant System (MMAS). Several studies of task allocation in grid environments have been proposed since the Max-Min ACO:

[10] uses the basic idea of MMAS ACO . The pheromone deposited on a trail includes a) an encouragement coefficient when a task is completed successfully and the resource is released, b) a punishment coefficient when a job failed and returned from the resource and c) a load balancing factor related to the job finishing rate on a specific resource.

[11] uses a balanced ACO which performs job scheduling according to resources status in grid environment and the size of a given job. Local pheromone update function updates the status of a selected path after job assignment. Global pheromone update function updates the status of all existing paths after the completion of a job.

[12] presents a metascheduler which decides where to execute a job in a service-oriented Grid environment consisting of several administration domains controlled by different local schedulers. The approach is based on the ant colony paradigm to provide good balance of the computational load. The information exchange protocol used is the Anthill framework [13] in which AntNests offer services to users based on the work of autonomous agents called Ants. A grid node hosts one running AntNest which receives, schedules and processes Ants as well as sends Ants to neighboring AntNests. State information carried by Ants is used to update pheromones on paths along AntNests. Additionally, Ants may carry jobs which have to be transferred and executed from one AntNest to another.

III. Scheduling Architecture

Regarding the system model, we consider a set of Computing Resource and a set of links. Each Computing Resource corresponds to a server, while each link corresponds to a physical link that interconnects two resources. Our system operates in a multi-tasking environment, i.e., several tasks may be executed on a single computing resource. We assume that computing resources are grouped to form an administrative domain. In essence, domains represent different network segments. A Grid resource provider may own and control one or more administrative domains in the Service Grid.

The ATS process, as a first step, requires a service that will act on behalf of the user submitting the tasks. Its role will be to capture the user preferences, requirements and constraints regarding the requested task and to deliver them in a suitable form to appropriate Grid resource provider entities. As a second step, ATS requires a service

that will act on behalf of a Grid resource provider. Each role will be to intercept user requests, acquire and evaluate the corresponding task requirements, computing resources and network load conditions, and ultimately, to select the most appropriate computing resource for the realization of the task. Furthermore, a monitoring service is required. The monitoring service consists of a distributed set of agents, which run on each computing resource. Each agent is responsible for monitoring the load conditions and available capacity of the computing resource and delivering them to the Grid resource provider related entity. Additionally, a distributed set of network provider related entities will be responsible for providing the Grid resource provider entity with network load conditions and managing the network connections necessary for resource provisioning.

OGSA covers the basic Grid functionality. The following extensions are made so as to cover the functionality that was identified above. First, the Scheduling Service (SS) is extended and assigned with the role of selecting on behalf of the Grid resource provider the best task scheduling pattern. Second, the Job Submit Service (JSS) is assigned with the role of promoting the user request for a specific task to the appropriate SS. Third, the Resource Agent (RA) is introduced and assigned with the role of promoting the current load conditions of a CCR. Finally, the Network Provider Agent (NPA) is introduced and assigned with the task of providing current network load conditions (i.e., bandwidth availability) to the appropriate SS. In essence, the distributed set of the RAs and NPAs forms the monitoring service.

```
<owl: Class rdf:ID= "TaskRequest">
  <owl:equivalentClass>
        <owl:Restriction>
           <owl:onPropeprty rdf:resource= "#hasJobType" />
           <owl:someValuesFrom rdf:resource= "#jobTypes" />
        </owl:Restriction>
        <owl:Restriction>
            <owl:onPropeprty rdf:resource= "#needsCPUs" />
            <owl:someValuesFrom rdf:resource= "#CPUSList" />
        </owl:Restriction>
        <owl:Restriction>
             <owl:onPropeprty rdf:resource= "#inputFiles" />
             <owl:someValuesFrom rdf:resource= "#inputSandbox" />
        </owl:Restriction>
        <owl:Restriction>
             <owl:onPropeprty rdf:resource= "#outputFiles" />
             <owl:someValuesFrom rdf:resource= "#outputSandbox" />
        </owl:Restriction>
        <owl:Restriction>
             <owl:onPropeprty rdf:resource= "#constraints" />
             <owl:someValuesFrom rdf:resource= "#constraintList" />
        </owl:Restriction>
  </owl:equivalentClass>
</owl: Class>
```

Table 1. TaskRequest class in OWL

Grid environments are highly dynamic and heterogeneous, characterized by the existence of diverse entities (resource requestors and providers) requesting and owning/controlling/maintaining various resources. Since successful task scheduling schemes depend on the quality of the available information [14][15], we adopt a Web Ontology Language (OWL) [16] semantic description for resources and tasks, so as to provide a common semantic understanding to be shared between the components involved in the scheduling process. As an example, task requirements are represented as objects of class TaskRequest which is described in OWL as shown in Table 1.

Figure 1 illustrates the grid services involved to the adaptive task scheduling process. The user submits a task request to the JSS which locates the SS and passes on the request. The SS locates through the Registry Service and contacts the service that represents the requested application resource , and acquires the requirements for the execution of the task. Consequently the SS queries the registry to find all CCRs and their respective capabilities, interacts with the RAs of the CCRs so as to obtain their current load conditions and with the NPAs so as to acquire the network load conditions. It ultimately evaluates the most appropriate task assignment pattern and informs the Application Resource to pass the task to the selected CCR for execution. Once the task terminates its execution, the Application Resource reports the result to the JSS, which, in turn, notifies the user.

Each SS controls the computing resources of a domain. Each RA service is associated with each computing resource, while each NPA service is associated with the network elements (e.g., switches or routers) necessary for supporting Grid Node connectivity. The RA, NPA role is to represent the computing resources or network elements, respectively, and to assist SS by providing information on the availability of resources of the computing resources/network element. Domain state information (load conditions of the computing resources of the particular domain and link utilization) is exchanged between the SS and the RAs/NPAs residing in the specific domain.

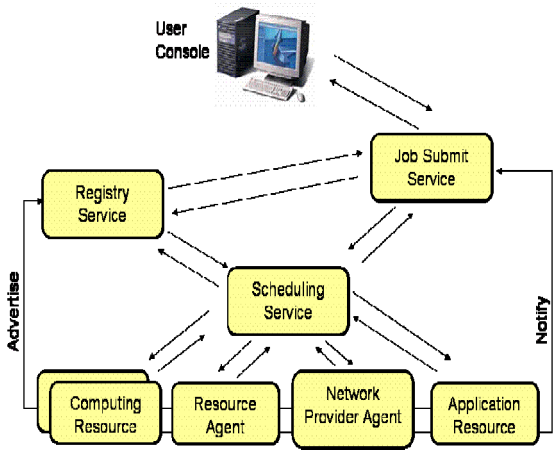

Figure 1. Grid Services providing Task Scheduling

The SS applies an extended version of the MMAS-ACO in which a three step framework is performed to select the suitable computing resource for each task. The first step includes the assignment of the task to the CCR with the maximum pheromone trail value. The second step applies a local search in order to improve the initial solution in case the system is congested and there are no resources with spare capacity so as to accommodate new task assignments. Finally, in the third step after task completion, the pheromone level on paths is updated using

the current optimal solution. Our approach is presented in a detailed manner in the next section.

IV. THE ACO ALGORITHM

In this paper an extended version of the MMAS ACO algorithm is used to solve the ATS problem. Each task is represented by an ant and the algorithm allocates ants to resources. The pheromone trail represents the desirability of assigning a task to resource. Solutions are constructed iteratively by assigning tasks to resources. After each iteration, the task deposits pheromone on the trail chosen. During the construction of a solution, ants chose the CCR where the task should be assigned. The initial pheromone value of each CCR is given by the formula:

$$\tau_j(0) = CPU\_Speed_j \cdot (1 - CPU\_Load_j) \quad (1)$$

Pheromone trails are updated upon assignment of a task on a CCR and upon termination of a task according to the formula:

$$\tau_j^{post} = \rho \cdot \tau_j^{pre} + \Delta\tau_{ij} \quad (2)$$

where τ_j^{post} and τ_j^{pre} are the trail intensity from a task to CCR j after and before the updating procedure. The excess in accumulation of pheromone is controlled by the use of the pheromone evaporation rate ρ (where $0 \le \rho \le 1$). The latter also allows the algorithm to avoid wrong decisions taken in the past [9]. $\Delta\tau_{ij}$ is the differentiation of the pheromone value a particular task i deposits on the path to CCR j. Specifically, the pheromone of a resource is reduced upon task assignment in relation to the task size and to the power of the resource. Adversely, the pheromone is refunded to the resource upon job termination. When task i is assigned to CCR j $\Delta\tau_{ij} = -M$, while when task i is completed and CCR j is released $\Delta\tau_{ij} = M$. M is a positive value relevant to the computation workload of the task. In the current version of this study, M is given by the following equation:

$$M = \frac{Task\_Instructions_i}{CPU\_Speed_j \cdot (1 - CPU\_Load_j)} \quad (3)$$

The factor $Task\_Instructions_i$ is the number of instructions task i contains. The CPU_Speed_j factor is the CPU speed of CCR j, while the CPU_Load_j factor refers to the CPU load of CCR j. In case CPU_Load_j approximates *1* the resource has not enough capacity and another assignment of a task on it should be avoided. The authors consider that $1 - CPU\_Load_j$ assumes the minimum value of 0.001, yielding thus a high value for parameter *M*, so that next tasks could not possibly be assigned on the specific resource. The desirability of assigning task i to CCR , is defined by the following formula:

$$des_{i,j} = \tau_j - Com\_Cost_{i.j} \quad (4)$$

Namely, a task i is assigned to CCR j in case $des_{i,j}$ takes the maximum value among all j resources. In this expression the factor τ_j is the current trail intensity on CCR j. The factor $Com\_Cost_{i,j}$ is the cost of migration to CCR j, plus the communication cost introduced in case i task needs to interact with other service components (e.g., other task or databases) residing on different resources to accomplish its goal. It is defined as:

$$Com\_Cost_{i,j} = \frac{Task\_Size_i}{Bandwidth_j} + \sum_k m_{i,k} \cdot cc_{j,l} \quad (5)$$

In the above formula the volume of messages exchanged between task i and component k for the accomplishment of task i is represented as $m_{i,k}$, and the communication cost per unit message that is exchanged between computing resources j,l is represented as cc_{jl} (we suppose that task i and component k reside respectively on j,l computing resources). This later factor may be proportional to the distance (e.g., number of hops) between the two computing resources and the load conditions (e.g., bandwidth availability) of the communication link interconnecting the two resources.

According to equation (4) the desirability value is proportional to the pheromone τ_j minus the communication cost given by formula (5). The authors have decided to include in their model the $Com\_Cost_{i,j}$, so as to have a more integrated solution of the task scheduling problem. The additional cost from the interaction of a task with other software units needs to be considered, since most current services are composed by distinct collaborative components.

Local search takes place when there are no resources with enough spare capacity. In such a case, the task is assigned to a resource according to the initial pheromone values of the resources ($\tau_j(0)$). Additionally, the initial pheromone values are also considered in case two or more resources have the maximum desirability value. In case of equal initial pheromone values, the task is assigned randomly to any of these resources.

V. EVALUATION

In this section, indicative results are provided in order to assess the proposed framework, which allows for adaptive task scheduling in Grid environments. In order to test the performance of the task scheduling scheme, we conducted experiments on a simulated grid environment composed of six computing resources with the following configuration: three computing resources with 3GHz CPU and 2 GB RAM, two computing resources with 3,2GHz CPU and 2 GB RAM and one computing resource with 2,7GHz CPU and 1 GB RAM. All computing resources reside on a 100Mbit/sec Ethernet LAN, running the Linux Redhat OS.

Concerning the implementation issues of our experiments, the overall Task Scheduling Mechanism has been implemented in Java. The Voyager mobile agent platform [17] has been used for the realisation of the services as well as for the inter-service communication. To be more specific, the system services (JSS, SS and the monitoring service RAs, NPAs) have been implemented as fixed agents and the task constituting the application

service as intelligent mobile agent, which can migrate and execute to remote computing resources.

To evaluate the efficiency of our task scheduling mechanism the following experimental procedure has been followed in a similar manner to [11]. We consider 2000 simple tasks, each performing matrix multiplication of real numbers. The matrix sizes are varying from *400x400* up to *1000x1000*. The task size depends on its matrix size and is about *n x n x 4* bytes (each real number is represented by 4 bytes). The number of instructions that the task contains, can be drawn from task's complexity. Since matrix multiplication has $O(n^3)$ complexity, $2n^3$ instructions are estimated for a $n \times n$ matrix multiplication. Since communication cost is similar for all hosts only the computation workload of tasks is considered.

In order to measure the efficiency of allocation schemes we use the standard deviation of CPU load of CCRs. The load of each CCR is taken after each task assignment and the standard deviation of each method is computed per 100 samples from 100 to 2000 tasks. The standard deviation is computed as:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

Where σ is the standard deviation, x_i is the CPU load of resource i and $\bar{x}$ is the average load of all resources. Low standard deviation values indicate that the system is better load balanced.

Initially we perform experiments to determine the most appropriate value of the evaporation rate coefficient ρ, when SS uses the ACO allocation scheme. Figure 2 illustrates the standard deviation of CCRs when the ACO task assignment scheme is used and ρ is set equal to 0.9, 0.8, 0.7, 0.6 and 0.5, respectively. As it can be seen, the lower standard deviation values are obtained when $\rho = 0.7$. This in essence means that with relative significance 70% the prior trail intensity from a task to CCR j is taken into account for the updating procedure, as ρ constitutes the memory of the system.

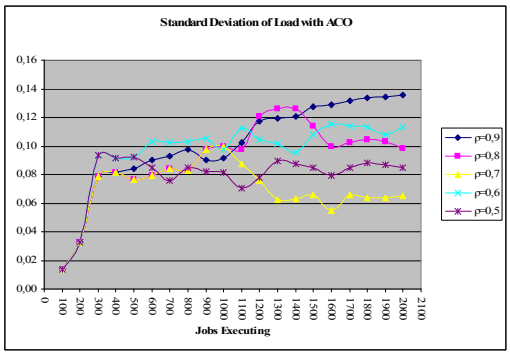

Figure 2. Standard Deviation of load for Ant Colony using various values for evaporation coefficient ρ.

Consequently, task assignments have been conducted with and without using the ACO task scheduling scheme. In the latter case, SS assigns tasks in a round robin fashion to computing resources.

Figure 3 illustrates the standard deviation of both methods. From the obtained results, we observe a decrease in the standard deviation when the ACO task scheduling scheme is used, which verifies that the load of CCRs is better balanced compared to the Round Robin assignment scheme. At this point it should be mentioned that the performance improvement is related to the number of tasks being executed on the Grid environment. It may be observed that for a small number of executing tasks (under 200 tasks) there is not significant improvement among the two different task assignment methods. This could be attributed to the fact that at this point the system is slightly loaded. However, in case more tasks exist in the system, ACO performs a lot better than simple allocation methods like Round Robin. Specifically the improvement introduced varies from 4.83% to 73.5%.

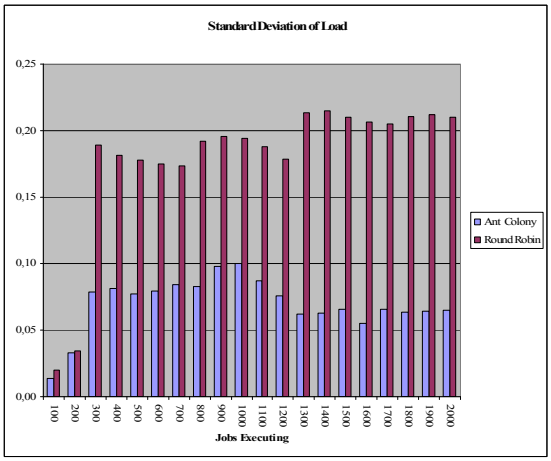

Figure 3. Standard Deviation of load for Ant Colony and Round Robin scheduling algorithms.

VI. Conclusions and Future Work

In this study the task scheduling problem in Grid computing environments has been addressed. Our objective is to design a scheduling architecture conforming to the OGSA standards, for Grid computing environments, providing the most appropriate assignment of tasks to computing resources, given the current load and network condition. The Ant Colony Optimization algorithm (ACO) was used to effectively assign tasks to computing resources. Our first experimental results indicate that the proposed framework produces good results.

Future work includes realization of further wide-scale trials, so as to experiment with the applicability of the framework presented herewith (e.g., consider the case of tasks requiring interactions with other resources so as to accomplish their goals) as well as the comparison of our scheme with alternative scheduling algorithms.

References

[1] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal of Supercomputer Applications, Vol 15 , I 3, pp. 200 – 222, Sage Publications, 2001.

[2] Global Grid Forum: http://www.ggf.org

[3] G. Coulson, P. Grace, G Blair, W. Cai, C. Cooper, D. Duce, L. Mathy, W. K. Yeung, B. Porter, M. Sagar, and W. Li, "A component-based middleware framework for configurable and reconfigurable Grid computing" Journal of Concurrency and Computation: Practice and Experience, Vol 18, I 8, pp. 865 – 874 John Wiley & Sons, 2005.

[4] J. Balasubramanian, D. Schmidt, L. Dowdy, and O. Othman, "Evaluating the Performance of Middleware Load Balancing Strategies", In Proc. of the 8th International IEEE Enterprise Distributed Object Computing Conference pp. 135- 146, Monterey, California, USA, (2004).

[5] A. Chavez, A. Moukas, and P. Maes "Challenger: A Multi-agent System for Distributed Resource Allocation" In Proc. of the 1st International Conference on Autonomous Agents, pp. 323- 331, New York, USA, 1997.

[6] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger "Economic models for resource management and scheduling in Grid computing" Concurrency and Computation: Practice and Experience, vol. 14, pp. 1507-1542, Wiley InterScience, 2002.

[7] D.A. Menascé, E. Casalicchio, and V. Dubey, "A heuristic approach to Optimal Service Selection in service oriented architectures", In Proc. Of the ACM Workshop on Software and Performance, pp. 13-24, Princeton, NJ, June 2008.

[8] Y. Gao, H. Rong, J. Z. Huang, "Adaptive grid job scheduling with genetic algorithms", Future Generation Computer Systems, Vol. 21, pp. 151–161, Elsevier, 2005.

[9] H. Lourenco, and D. Serra, "Adaptive search heuristics for the generalized assignment problem" Mathware and Soft Computing, Vol. 9, pp. 209–234, 2002.

[10] H. Yan, X. Qin, X. Li, M. Wu, "An improved ant algorithm for job scheduling in grid computing", In Proc. of the 2005 International Conference on Machine Learning and Cybernetics, vol. 5, 18–21 pp. 2957–2961, Guangzhou, China, 2005.

[11] R. S. Chang, J. S. Chang, P. S Lin, "An ant algorithm for balanced job scheduling in grids" The International Journal of Grid Computing: Theory, Methods and Applications, Vol. 25, pp. 20–27, Elsevier, 2009.

[12] K. Dörnemann, J. Prenzer, and B. Freisleben, "A peer-to-peer meta-scheduler for service-oriented grid environments" In Proc. of the first international conference on Networks for grid applications (article no 7). Lyon, France, 2007.

[13] O. Babaoglu, H. Meling, and & A. Montresor, "Anthill: A framework for the development of agent-based peer-to-peer systems", In Proc. of the 22th International Conference on Distributed Computing Systems, pp. 15–22, Vienna, Austria. IEEE, 2002.

[14] P. Missier, P. Wieder, and W. Ziegler, "Semantic Support for Meta-Scheduling in Grids", Knowledge and Data Management in GRIDs, pp. 169-183, Springer US, 2007.

[15] A. C. T. Vidal, F. J. S. Silva1, S. T. Kofuji and F. Kon, "Applying semantics to grid middleware", Concurrency and Computation: Practice and Experience, Vol 21, pp. 1725-1741, 2009.

[16] OWL Web Ontology Language Overview: http://www.w3.org/TR/owl-features/

[17] Voyager Platform, Recursion Software Inc. http://www.recursionsw.com/