Journal of Information Technology and Applications
Vol. 1 No. 4 March, 2007, pp. 249-260

An Ant Colony-based framework for Internet-scale Peer-to-Peer Grids

Francesco Palmieri
CSI, Federico II University - Via Cinthia 5, 80126 Napoli, Italy
Francesco.Palmieri@unina.it

Abstract

Grid is an exciting buzzword in the computing world today, mainly in the scientific area. It is usually defined as the exploitation of a varied set of networked computing resources, including large or small computers, PDAs, file servers and graphics devices. The emerging Internet based peer-to-peer Grid infrastructures, which are based on a "flat" organization allowing seamless discovery, access to, and interactions among resources and services, have complex and highly dynamic computational and interaction behaviors, and when combined with the uncertainty of the underlying Internet infrastructure, result in significant development and management challenges. Accordingly, in this paper, we propose a multi-agent based Peer-to-Peer Grid computing architecture based on swarm intelligence and precisely on the ant-colony meta-heuristic, to map the solution capability of social insects to the peer node interaction and resource scheduling problem. The main characteristics of this architecture are highlighted by its promising performance and scalability, and its adaptive resource management and scheduling mechanisms. It is completely decentralized, allowing every node in the system to act as a client and a runtime server, by generating new jobs, submitting them to the network and participating, by running several jobs, to the overall computational resource sharing. A Grid infrastructure designed in this way may be interesting for individual peer entities that want to share their resources through the Internet within a large community in order to exploit the resulting computing power effectively. Neither class of current Grids is capable of scaling up to a so large user base, with a wide variety of applications and a plethora of user profiles. This will force distributed application designers to look for alternative, more suitable computational models and architectures.

Keyword: Peer-to-Peer Grids, Ant Colony Optimization, Multi-Agent Systems

1. Introduction

In today's incredibly sophisticated world of computation, with the emerging very high speed machine processing capabilities, complex data storage methods, evolutionary operating systems and services, and extremely advanced optical networking services capabilities - we are entering a new era of truly distributed computing. A growing number of high-performance scientific and industrial applications, ranging from real-time particle physics or radio astronomical experiments to complex weather forecast, and financial modeling, are increasingly taking advantage from large geographically distributed computing, network infrastructures and data management resources, commonly referred to as "Grids", named after the analogy with electricity grid. Like an electrical power grid, the Grid will aim to provide a steady, reliable source of computing power. A Grid also offers a uniform and often transparent interface to its resources such that an unaware user can submit jobs to the Grid just as if he/she was handling a large virtual supercomputer, so that large computing endeavors, consisting of one or more related jobs, are then transparently distributed over the network on the available computing resources, and scheduled to fulfill requirements with the highest possible efficiency. The emerging domain of Grid computing solutions has yielded a wide variety of network infrastructures with very different capabilities and features. At one extreme, based on transparent optical communication and Wavelength Division Multiplexing technologies, we can find very high performance, dedicated fiber-based infrastructures, such as the lambda-Grid [1] that is able to couple traditional distributed Grid resources with reserved lambdas or wavelengths, with 1-10 Gbps per link, and hundreds of lambdas per optical fiber. At the other extreme, we can find another class of Grid infrastructures, called Peer-to-peer Grids, inspired by the cycle sharing peer to peer applications, where the underlying transport infrastructure is the standard ubiquitous Internet, offering no specific support for the Grid applications, and often resulting in a rather poor network efficiency and conflicts with "normal" internet traffic. The greatest enabling factor for peer-to-peer Grid architectures is the widespread availability of high-end desktop PC or Workstation always connected to the Internet that at the state of the art offer a computational capacity of 3-5 GFlops, that is expected to become in the order of 100 GFlops within the same time frame. Such a great processing power that makes it possible to execute extremely demanding applications (at least by today's standards) is largely unused (at least for the most part of the day). This opens up a very interesting window for computational resource sharing, also sustained by the current trend of growth of the bandwidth availability

on the average and high-end Internet connections (doubling each year) making ubiquitous Internet-based peer-to-peer Grid computing one of the most valid options available in the computing arena. For example, in the Seti@Home project [2], the enormous amount of radio signals registered by radio telescopes are subdivided into a large number of data sets, that can be independently distributed through the Internet and analyzed in the search for evidence of extra-terrestrial intelligence; Distributed.net [3] is an umbrella for several distributed computing projects, including cryptography challenges in which brute-force attacks are performed by subdividing key spaces into independent portions; the Anthrax Project [4] is an effort designed to help scientists to find a treatment for the Anthrax toxin, by performing screening analysis of large sets of molecules.

For the majority of grid systems and particularly for peer-to-peer Grids, scheduling of customer jobs on the available computational resources is one of the most important topics. In the simplest case, scheduling of jobs can be done in a blind way by simply assigning the tasks to the compatible resources according to their availability. Nevertheless, it is much more profitable to use more advanced and sophisticated distributed scheduling policies capable to react to the dynamics of the grid system, typically by evaluating and fairly balancing the current load of the Grid resources, for better coping with unexpected peaks of activity. Computing tasks are assigned to each resource available on the peer-to-peer Grids through a typical request/grant cycle handled by job schedulers (also called grid resource brokers) running on specific "master" nodes. These job schedulers attempt to distribute the demands across the Grid as optimally as possible, based on application profile and resource availability. However this can be viewed essentially as a master-slave architecture, in which only few properly designated master nodes, often becoming bottlenecks in very large environments, are enabled to assign new jobs and the large amount of slave machines distributed on the Internet are relegated to a role of mere executors, thus in some sense betraying the peer-to-peer philosophy.

On the other side, the communication pattern between the components of the distributed applications is also highly differentiating in Grid environments. The advent of large-scale computation and data sharing in wide-area Grids and peer-to-peer applications is driving an evolution in communication patterns from point-to-point connections to multipoint-to-point and multipoint-to-multipoint structures. Examples include many widely deployed Peer to Peer content delivery networks, in which multiple replicated data repositories are simultaneously accessible for remote visualization or computation at much higher aggregate speed.

Starting from the above considerations, this paper proposes a novel multi-agent based Grid paradigm based on the concept of real Peer to Peer resource sharing by direct exchange between peer nodes (i.e., nodes having same role and responsibility) aiming to extend distributed computation through the Internet to the general public. Distributed information exchange, needed for scheduling and control will be based on swarm intelligence [5] that is the collective behavior from a group of social insects, namely ants, that communicate interactively either directly or indirectly in a distributed problem-solving manner. Such behavior can be easily modeled as a multi-agent system where all the agents communicate each other through the network and cooperate through stigmergy according to ant-like multipoint-to-multipoint local interactions. The intrinsic nature of multi-agent technology, explicitly oriented to model high dynamic and complex systems [6], seems to be well suited to provide support to peer-to-peer Grid computing needs. Moreover, the adoption of agent technology could bring to Grid users and administrators more friendly and understandable interfaces to interact with the system. The resources that can be exchanged through this new Grid infrastructure would include content, as in popular Peer to Peer file sharing applications, and storage capacity or CPU cycles. The peer nodes will operate as both clients, by submitting jobs to the Grid, and runtime or management servers, by sharing their computational power and participating through swarm intelligence to the resource scheduling and Grid management. The proposed architecture, while rather complex from the control logic point of view will introduce some undeniable benefits in the overall Grid infrastructure. First, it will allow distributed applications to reach out to harness the outer edges of the Internet and consequently will involve scales that were previously unimaginable. Second, being inherently based on the peer to peer paradigm by definition, it excludes any form of centralized structure, requiring scheduling and control to be completely decentralized. Finally, and most importantly, the environments in which these new Grid applications will be deployed exhibit extreme fault tolerance, dynamism in structure and load. Extensive simulation results obtained upon different experimental Grid topologies clearly indicate that our ant-colony based approach is highly adaptive, robust and effective in handling the distributed resource discovery and scheduling in Internet-scale peer-to-peer Grids.

2. A new approach to Grid computing

The traditional and peer-to-per Grids have been analyzed to examine their strengths and weaknesses.

Then an evolutionary approach with all its basic theoretical concepts and building blocks can be proposed to help in addressing some of these concerns, enabling seamless integration of distributed computing systems allowing decentralized control, large-scale and extreme dynamism in the peer-to-peer Grid environment.

2.1 Traditional vs. peer to peer Grids

While computing Grids have been widely used in computational science, Peer-to-Peer computing has achieved wide prominence in the context of multimedia file exchange. It uses the computing power at the edge of a connection rather than within the network. The client/server architecture does not exist in a peer-to-peer system. Instead, peer nodes act as both clients and servers - their roles are determined by the characteristics of the tasks and the status of the system. This architecture minimizes the workload per node, and maximizes the utilization of the overall computing resources among the network. Today, the shear numbers of desktop systems make the potential advantages of interoperability between desktops and servers into a single Grid system quite compelling. However, the peer-to-peer systems have significantly different properties than the conventional client/server-based Grid systems. The modern peer-to-peer Grid infrastructures attempt to collect resources from a variety of providers that are managed in a totally distributed fashion, and are able to handle from hugest to relatively small jobs. Their strength lies in the capability of offering a generic service based on a large number of distributed resources, thus able to allow a wider variety of applications. Conceptually, as all the known Peer-to-peer systems these new computing infrastructures are characterized by decentralized control, large-scale and extreme dynamism of their environment. They are usually highly autonomous and heterogeneous systems and their availability varies from time to time. Their main requirement becomes the Internet-wide scalability for all the control and management services such as resource management, co-allocation, and scheduling.

On the other side, the traditional approaches, very common in early production grids, are based on a service-oriented computing architecture with a super-local resource management and scheduling strategy. In detail, the overall control logic is based on a certain number of centralized managers, often called resource brokers, that are the only entities with a complete view of the resources available on the whole Grid or on their own local management domain. Each broker selects computing resources based on actual job requirements and a number of criteria identifying the available resources and their location with the aim to minimize the total time to delivery for the individual application, and performs job distribution on them. This is clearly not applicable in modern peer-to-peer grid computing where both the network and the computing infrastructure itself lack of a fixed structure. In fact the service oriented architecture has poor adaptability in terms of performance, availability, and scalability, since no facility has been provided by the Grid control systems to allow automatic deployment of the services according to the clients' requests and the load of the Grid. The dependence on the schedulers increases the complexity of application programming in the Grid environment, as it is difficult to provide the various local schedulers with a uniform programming interface that supports task decomposition, state persistence, and inter-task communications. The overall resource management and scheduling strategy intensively relies on the client-server hierarchical approach. This two-level process leads to more complex handling on resource discovering, selection, and allocation compared with a one-level process. The lack of a distributed management policy of the computing nodes can cause unsuitable selection of resources, and unbalanced loads, and therefore limits the overall performance. Consequently, a peer-to-peer interaction strategy between the Grid nodes, operating in a "flat" organization, seems to be the most promising approach to provide scalable and adaptive services for the next generation Grids.

2.2 Ant-based peer to peer organizations

A key issue in the development and management of the peer-to-peer Grid infrastructures is the coordination of the distributed autonomous computing elements. The defining characteristics of these emerging systems include:

- heterogeneity, Grid environments and applications aggregate large numbers of computational and information resources;
- dynamism, the computation, communication and information environment is continuously changing during the lifetime of an application, including the availability and state of resources and services;
- uncertainty caused by multiple factors, including dynamism, which introduces unpredictable and changing behaviors that can only be detected and resolved at runtime, failures, which have an increasing probability of occurrence as the system scales increase, and incomplete knowledge of global system state.

Furthermore, complex job scheduling and composition/decomposition should be supported since in many cases a single available element might not address specific application requirements, and composing several cooperating elements to form a

unit with integrated computing functionalities is necessary to support the most computational power-hungry applications. The resource discovery, paradigms together with such scheduling and job composition/decomposition criteria, enable applications or parts of them to be dynamically executed on a virtual computing environment composed from discrete elements to meet the changing requirements and system behaviors, deal with element failures, optimize performance, address QoS constraints, etc. However, such a distributed coordination strategy is very difficult in large Grid environments where all the computing elements are available in a totally dynamic on-demand manner. Challenges include element descriptions, their discovery, and their dynamic and adaptive composition, interaction and coordination. Thus the critical components of an implementation infrastructure for autonomic composition include an efficient, scalable and flexible discovery mechanism, and a high-level integration mechanism. The discovery mechanism enables the selection of appropriate elements while the integration mechanism enables selected elements to be composed coherently, without conflicts in element dependencies and interactions. At our advice, autonomic self-organization strategies inspired by biological systems, as embraced by the modern multi-autonomous agent technology and the ant-colony meta-heuristic intelligence, aiming to map the solution capability of social insects to the above distributed scheduling and management scenario, seem to be the most promising approaches to address these challenges. More specifically, the simple observation of the phenomenon of a group of ants in a natural environment, which can dynamically and adaptively find and collect foraging objects into their nest without any master or central authority driving them, gives a significant clue to solve our distributed control and scheduling problem. The ants work together to achieve an optimal solution and move towards the optimal solution by sharing their own knowledge with their neighbors. Such behavior can be easily modeled as a multi-agent system where all the agents communicate each other through the network and cooperate through stigmergy according to ant-like local interactions. Each task is carried by an ant, realized by an agent. Ants cooperatively search the less-loaded nodes with sufficient available resources, and transfer the tasks to be executed to these nodes. The proposed self-organizing mechanism does not need a centralized control, which otherwise might act as a potential bottleneck, and is an attractive solution for very large, dynamic and computationally intensive Grid infrastructures because it is inherently parallel and easy distributable, with each node running one or more agents performing search in the solution space or directly managing the resources available on it. Adding more agents (or “ants”) generally increases the solution quality at the cost of a very limited additional workload

2.3. Multi-Agent Systems

A software agent is an autonomous software entity able to expose a flexible behavior. Flexibility is obtained by means of reactivity, pro-activity and social ability [7]. Reactivity is the ability to react to environmental changes in a timely fashion while pro-activity is the ability to show a goal directed behavior by taking the initiative. Due to their reactivity agents are also very reliable components to build fail safe systems, since their autonomous behavior easily allow recovering from fault conditions. Social ability, that is the ability to interact with peers by means of cooperation, negotiation, and competition, is one of the most important features of agent oriented programming: agents do their best when they interoperate. Interaction is obtained by arranging agents in communities called multi-agent systems. The intrinsic nature of Agent technology, explicitly oriented to model high dynamic and complex systems [6], seems to be well suited to provide support to Grid computing needs. In particular, agents can play many different roles into Grid organization, be organized into dynamic groups, and be able to migrate between nodes and groups to support scheduling and load balancing.

2.4. The Ant Colony meta-heuristic

The ant colony optimization (ACO) techniques are a subset of swarm intelligence meta-heuristics inspired by the foraging behavior of real biological networks [8] in finding the paths to food sources and route around obstacles and consider the ability of simple individuals to solve complex problems by cooperation. In biological “networks” or colonies of ants consisting of thousands and in some cases tens of thousands of dynamic elements, each ant alone has relatively little intelligence, while the collective emergent behavior of the “network” exhibits a great deal of global intelligence capable of dynamic near-global optimization of certain tasks. Engineering models and algorithms based on these biological systems have the potential to leverage the tremendous gains made in this century in understanding their individual and collective colony-based behavior. Initial work in swarm intelligence has revealed a great deal of synergy between the routing requirements of communication networks and certain tasks that exist in biological swarms. For instance, a key characteristic of swarm intelligence is the ability of search agents (ants) to find optimal (or near optimal) routing (in food gathering operations for example), where intelligent behavior arises through indirect communications between the agents. The

latter point is the most interesting: the ants do not need any direct communication for the solution process, instead they communicate by stigmergy. The notion of stigmergy means the indirect communication of individuals through modifying their environment. In detail, any ant that leaves its colony in search of food leaves a trail of chemical called pheromones on the path that it takes. When an ant returns from the food source to its nest, it reinforces the pheromones on the path that it has used. Pheromones acts to attract other ants to follow a particular path. When a large number of ants forage for food, the shortest or however the best available path to the food source will eventually contain the highest concentration of pheromones, thereby attracting all the ants to use that path. Thus, the concentration of pheromone on a certain path is an indication of its usage. With time the concentration of pheromone decreases due to diffusion effects. This property is important because it is integrating dynamic into the path searching process. The following figure shows a scenario with two routes from the nest to the food place. At the intersection, the first ants randomly select the next branch. Since the upper route is shorter than the lower one, the ants which take this path will reach the food place first. On their way back to the nest, the ants again have to select a path. After a short time the pheromone concentration on the shorter path will be higher than on the longer path, because the ants using the shorter path will increase the pheromone concentration faster. The shortest path, that in this case is straightforwardly the best one, will thus be identified and eventually all ants will only use it.

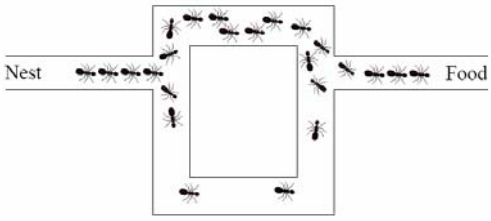

Figure 1. The ant path searching behavior

Algorithms modeled according to this behavior, have historically been used to solve shortest path problems and problems that can be reduced to a kind of shortest path problems but have also been successful in attacking various popular combinatorial optimization problems such as the traveling salesman problem (TSP), the quadratic assignment problem (QAP) and the job-shop scheduling problem (JSP) [9]. Furthermore, almost all the results obtained to date in developing swarm-based path searching or combinatorial optimization algorithms exhibits the following potential benefits:

- Dynamic "online" optimization using local information
- No or very limited exchange of global information for solution determination
- Inherent scalable nature, resulting in graceful builds and degradations
- Characteristics leading to adaptivity and robustness under most contingencies

The solution to a combinatorial optimization problem is a set $S = \{c_1, c_2, \ldots c_n\}$, where c_i, with $1 \le i \le n$, are known as the solution components. The Ant Colony optimization techniques are inherently iterative in their behavior. During each iteration, an ant constructs a solution starting from the empty solution $S = \varnothing$. Solution components are incrementally added, one at a time, to the partial solution until a complete, feasible solution is formed. At the end of an iteration, the candidate solutions constructed by the ants will be evaluated using a problem specific objective function f. After that, each ant will update the artificial pheromone associated to each component that is found within its constructed solution. Components that appear frequently in good candidate solutions will have higher pheromone values. The search process in subsequent iterations will then be biased towards these favorable components. The search process can stop if for example the number of maximum iterations has been met. In any case, the five main characterizing elements of the above techniques are identified below:

- A heuristic function η, which will guide the ants' search with problem specific information.
- A pheromone trail definition, which states what information is to be stored in the pheromone trail. This allows the ants to share information about good solutions.
- The pheromone update rule, this defines the way in which good solutions are reinforced in the pheromone trail.
- A fitness function which determines the quality of a particular ant's solution.
- A construction procedure that the ants follow as they build their solutions (this also tends to be problem specific).

Although the ACO method is a fairly novel approach, it can be viewed as something of an amalgam of other techniques. The ants essentially perform an adaptive greedy search of the solution space. Greedy search methods have been used for many problems, and are often good at finding reasonable results fast. The greedy search takes into account information provided in the pheromone trail, and this information is reinforced after each iteration. Because a population of ants is used, ACO also has similarities with evolutionary approaches such as

genetic algorithms (GAs). GAs use a population of solutions which share useful information about good ones, and probabilistically use this information to build new solutions. They do this in a rather different way, however, since a GA "stores" its information in the population at each iteration whereas the ants store it in the pheromone trail. So, while the specific details of an ACO algorithm are new and the techniques used are combined in a novel way, the ACO approach implicitly uses several proven problem solving strategies.

3. Architectural details

In a traditional Grid architecture there are two kinds of participating entities: the clients submitting computing tasks to the Grid and the computing nodes (often called servers). A client is implemented as a generic computing device that requests services from the Grid by using properly designed web interfaces (service portals) and middleware services, usually implemented through the Web Services standards, that assist grid users in finding suitable resource to run their jobs. A computing node is the place where tasks are executed and computing occurs. Each computing node runs the runtime and management environment for jobs and services.

In our peer to peer model a generic computing device participating to the Grid can serve as a client and a peer at the same time. There are no super-local resource brokers or privileged nodes and the management of all the grid resources requires new and smarter technologies to implement a totally distributed and highly dynamic scheduling environment to allow scalable, failure-safe and improved resource utilization. Accordingly, an ant-like self-organizing mechanism can be used to perform efficient resource management on the Grid nodes through a collection of very simple local multipoint-to-multipoint interactions. This can be achieved by heuristically determining a scheduling solution that distributes the jobs on the Grid resources minimizing the overall Grid *Makespan*, that is the maximum completion time of all the job instances in the schedule, and *Flowtime*, referring to the response time to the user petitions for task executions. The above process can result in an indirect coarse-grained load balancing effect since each task tends to be dispatched to a grid resource that has less workload and can meet the application execution deadline.

3.1 The agent-based distribution model

The multi-agent technology has features well fitting for distributed communication, and is particularly robust for the interaction, negotiation and scheduling processes. We consider that such characteristics are important fundamentals for a suitable solution to the problem in question. In a distributed agent framework, we conceptualize a dynamic community of agents, where multiple agents contribute services to the community by cooperating like individuals in a social organization, such as an ant colony. The appeal of such architectures depends on the ability of populations of agents to organize themselves and adapt dynamically to changing circumstances without top-down control from a central control logic. Although agents have been applied to computing load balancing for many years, only some attempts to apply intelligent agents in realizing the Grid vision have been made by academic researchers in the last few years. Anyway, the agent's social ability, the autonomous and flexible behavior could play an important role for the communication and the interaction with different nodes, for example, in the exchange of information about the resources available on each node.

3.2 Management and discovery

The proposed resource management and discovery framework for peer-to-peer Grids can be implemented as a multi-agent system where each agent is a representative of a local grid resource (resource management agent) or of an independent execution request from an user (search agent), associated with explicit computational requirements. Each node on the networks supplies the same service and each search agent must be served by one node through one of its resource management agents. Each resource management agent, that must be present on each node participating to the peer-to-peer grid, is used to manage all the applications within "one" computing resource which could be a PC, a workstation, or a cluster of computers, and is responsible to schedule this computing resource. This kind of agent should have knowledge about the hardware performance of this computing resource, and all applications installed with it as well as the performance of these applications in this particular hardware environment. Jobs are submitted to the grid through the resource management agents that cooperate to identify the candidate resource (computing node) on which the task will be executed. When a job is received by the resource management agent, a job-related search agent will be created. This agent will be in charge of finding candidate resource management agents through the proposed ACO-based interactions, and finally selecting the best resource agents to execute the job. The search agent lives until the associated job executes and will be dissolved when the job is finished and the results are sent to the end user (or may also be saved in a secure storage resource). The search agent must be resource-aware and be able to express its resource needs to the system, and be able to negotiate for system resources. Once scheduled, the search agent must also be notified if the system state has changed in a way that

impacts the job (computing tasks) and perhaps is necessary to enter resource re-negotiation and job re-scheduling. At this point, the search agent may need to adapt job execution to a different set of allocated resources. It should be noted that a resource agent or a search agent might be involved in several negotiations of different types at the same time, which is usually called combined negotiation. Combined negotiation is another difficult issue, particularly in a multi-agent system implemented with the proposed adaptive negotiation approach. Since cooperation, negotiation, and competition are natural activities common in multi-agent systems this functionality is naturally obtained by using the agent oriented approach. In the same way localization of services and coordination within a single node are obtained with less effort. All the agents, that in our ACO-based model can be viewed as the "ants" are generated from nodes on the grid according to their specific roles, that is, the system consists of a certain number of ants/agents which either handle computational resources in various nodes or wander on the network, searching for available resources. The search agents associated to a task not yet assigned to a computing resource for execution (pending task) wander on networks driven by the ACO meta-heuristic and search proper nodes (resource management agents) to join and queuing by interacting with the corresponding resource management agents. At an higher level of the peer-to-peer Grid control logic, agents cooperate with each other according to ant-like local interaction to manage the overall workload according to the above agent-based self-organization performing complementary scheduling of the available resources on the network-distributed Grid.

3.3 The job scheduling logic

In our model we consider, for simplicity sake, the following basic assumptions in formulating the ACO-based scheduling logic: the jobs being submitted to the grid are independent and are not preemptive, that is, they cannot change the resource they has been assigned to once their execution is started, unless the resource is dropped from the grid. Examples of this scenario in real life grid applications arise typically when independent users send their tasks to the grid, or in case of applications that can be split into independent tasks. Such applications are frequently encountered in scientific and academic environments. They also appear in intensive computing applications and data intensive computing, data mining and massive processing of data, etc.

Our objective is to determine an efficient solution to the problem of scheduling a set J of m independent jobs $J=\{j_1, \ldots j_m\}$ each of which have an associated predicted running time $t(j_i)$ onto a set V of n Grid nodes, $V=\{v_1, \ldots v_n\}$, for each $1 \le i \le n$, with a specific computing capacity C_i, such that the load will be fairly balanced on the available computing resources and consequently all the jobs are completed as quickly as possible, by optimizing the overall Grid performance. Really, this is a multi-objective optimization, the two most important objectives being the minimization of the makespan and the flowtime of the overall grid system. This problem, widely simplified, is closely related to a commonly known NP-hard combinatorial optimization problem, the bin packing problem, where the items to be packed are viewed as the jobs and the bins as the Grid nodes with their capacity represented by the available computing power. Consequently, also our multiple variable resource scheduling problem, that is inherently more complex, will be NP-hard and thus a reasonable heuristic solution, achieving near-optimal results is strongly desirable. Accordingly we propose an highly adaptive approach that employ a collection of ant agents that collaborate to explore the search space. A stochastic decision making strategy is proposed in order to combine global and local heuristics to effectively conduct this exploration. As the algorithm proceeds in finding better quality solutions, dynamically computed local heuristics are utilized to better guide the searching process. In our schema an ant's "life" begins at the originating node of each task execution demand. It proceeds until it finds the corresponding destination node resource management agent offering adequate resources available for task execution. At the completion of its search, each ant deposits pheromone to mark the detected solution and perform its stigmergy-based interaction with the other socially-related individuals. Thus, the most important factor that influences each ant's decision, and hence the overall heuristic search behavior, will be the pheromone.

3.3.1. The pheromone trail

There is not an immediately clear definition for the pheromone trail in this problem, and so the information that will be stored in the pheromone trail must be carefully selected. As the problem is essentially to allocate jobs to the grid nodes, intuitively it seems that the trail could store the degree of success associated to assigning a particular job to a particular node. The pheromone value $\tau(i,j)$ can be therefore selected to represent the overall success in finding a feasible solution implied by scheduling a particular job i onto a particular node j. The pheromone matrix will thus have a single entry for each job-node pair in the problem. Furthermore, a policy for updating the pheromone trail has been established according to the following equation:

$$\tau(i,j) = \rho \cdot \tau(i,j) \qquad (1)$$

where ρ is a parameter which defines the pheromone evaporation rate and decays the pheromone trail implementing a mean of 'forgetting' solutions which are not reinforced often.

3.3.2. The ant-driving heuristic

The heuristic information that the ants use when building their solution is also very important, it guides their search with problem specific information. However, because the ACO approach relies on multiple ants building solutions over several "generations" the heuristic information must be quick to establish, and so only fairly simple heuristic values can be used. The heuristic value used by the ants for each job *j* has been defined in our model as:

$$\eta(j) = \frac{1}{\min_i(\log_{10}(t(j_i)))} \qquad (2)$$

That is inversely proportional to the minimum completion time for the job j on all the available nodes, or better stated its completion time on the best available node on the grid. To allow this value to be effectively controlled with the β parameter, determining the extent to which heuristic information is used by the ants, it is necessary to "scale" the heuristic value up. Therefore in the implementation of this function all the $\eta(j)$ values are computed for each job and then the job list is sorted into descending order of these values.

3.3.3. Finding a solution

The set of jobs and nodes will serve as components from which each ant will use to incrementally construct a solution during each iteration of the algorithm. For each ant, an iteration consists of a finite number of steps. At step *r* of iteration *t*, an ant *k* will select a specific job *j* to be executed on a node v_i to be included in its partially constructed solution $Sk(t,r)=\{s1,\dots sj\}$ according to a stochastic decision making strategy properly driven by the above heuristic and pheromone trail. More precisely, let b_j the node on which the job *j* can be executed in the minimum completion time, and α the extent to which pheromone information is used as the ants build their solution, the probability $p(j)$ of selecting job *j* to schedule next is given by equation:

$$p(j) = \frac{\tau(j,b_j)^{\alpha} \cdot \eta(j)^{\beta}}{\sum_{i=1}^{n}(\tau(i,b_i)^{\alpha} \cdot \eta(i)^{\beta})} \qquad (3)$$

A job is then selected based on this value, and the chosen job j_c is preferably allocated, on the b_{jc} node. The pheromone trail update procedure is then used on the iteration best ant. This process is repeated until all jobs have been scheduled, a complete solution has been built and there is no further improvement in the fitness value of the solution. The fitness value is determined according to a properly-crafted fitness function, whose goal is essentially to help the algorithm in discerning between high and low quality solutions that in our case means obtaining that all the jobs are completed as quickly as possible. In other words it implicitly means best balancing the load on the available nodes, that is, minimizing the makespan and flowtime of the solution itself. Makespan and flowtime are both strong fitness indicators of the grid system; their relation is not trivial at all, in fact they are contradictory in the sense that trying to minimize one of them could not suit to the other, especially for plannings close to optimal ones. Note that makespan is an indicator of the general productivity of the grid system. Small values of makespan mean that the scheduler is providing good and efficient planning of tasks to resources. On the other hand, minimizing the value of flowtime means reducing the average response time of the grid system. For better results, the value of mean flowtime, flowtime/*M* (where *M* is the number of machines in the grid), can be used instead of flowtime. Essentially, we want to maximize the productivity (throughput) of the grid environment through an intelligent load balancing and at the same time we want to obtain plannings that offer a quality of service acceptable to the users. Consequently the fitness function for our assignment and balancing problem could simply be the inverse of the sum of makespan mks_s and mean flowtime ft_s of the solution *s*, weighted by a properly crafted parameter λ, a priori fixed to 0.75 to give more priority to makespan, as it is the most important parameter. The fitness equation is reported below.

$$f_s = \frac{1}{\lambda \cdot mks_s + (1-\lambda) \cdot ft_s} \qquad (4)$$

The presented ACO-driven scheduling algorithm is conceived to be flexible in the sense that the number of ants can be adjusted by the launching probability of ants to achieve a good performance.

3.4 The agent structure

In our environment, the specific agents described in the previous section will be implemented at the Grid middleware level both for managing computing resources on each local grid node and scheduling incoming tasks, and for discovering available resources on the network. Agents provide an high-level representation of the corresponding grid capability. They also characterize the available resources as high performance computing service

providers in a wider grid environment. According to the convergence of the three major technologies for distributed systems - Grid, Agents and Web Services - under the umbrella of the Open Grid Service Architecture (OGSA) [11], each agent will be implemented as a service that conforms to the set of conventions for Web Services. Web Services have emerged as a set of open standards, defined by the World Wide Web consortium, and ubiquitously supported by Information Technology suppliers and users. A Web Service interacts with its environment through a collection of operations that are network-accessible through standardized XML messaging. A Web Service is described by an XML-based service description that covers all the details necessary to interact with the service, including message formats, transport protocols and location. For an application to take advantage of Web Services, three operations have to take place: publishing service descriptions, looking up service descriptions, and binding (or invoking services) using such service descriptions. They rely on the above XML syntactic framework, the transport layer SOAP [12], the XML-based language WSDL [13] to describe services, and the service directory UDDI [14]. In detail, all the agent ontologies will be defined using XML Schema components, and the agent behavior will be described as a WSDL interface. The benefit is that by publishing agents as service descriptions, other Web Services may make an effective binding and dynamic invocation of the agent seen as a Web Service, regardless of whether it is an agent-based computing functionality behind.

Agents can be structured according to a layered model to better define their architectural characteristics and ease the overall implementation tasks, as defined in the following schema [10].

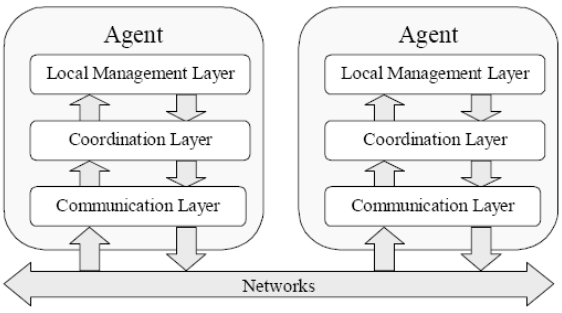

Figure 2. The agent multi-layer architecture

- *Communication Layer.* Agents in the Grid system must be able to communicate with each other or with users using common data models and communication protocols. The communication layer provides an agent with an interface to heterogeneous networks and operating systems. Agent-agent interaction is exclusively via SOAP message passing. Asynchronous message passing has good scalability characteristics with a minimum of synchronization between the agents.
- *Coordination Layer.* The request an agent receives from the communication layer should be explained and submitted to the coordination layer, which decides how the agent should act on the request, during the search space exploration, according to its own knowledge. For example, if an agent receives a service request, it must decide whether it has related service information. The ant-driven resource scheduling process does not aim to find the best service for each request, but endeavors to find an available service provided by a neighboring agent. While this may decrease the optimal load balancing effect, the trade-off is reasonable as grid users prefer to find a satisfactory resource as fast as possible.
- *Local Management Layer.* This layer performs functions of an agent for local and grid resource management that is participating to scheduling decisions to handle load distribution between the local resources (i.e. local processors) and the overall Grid resources. A local grid resource is considered to be a cluster of tightly coupled workstations, i.e. in a blade enclosure, operating according to a common scheduling policy, or a multiprocessor system. An agent takes its local available computational resources as one of its capabilities. It is also responsible for submitting local service information to the coordination layer for agent decision making. Within each agent, its own service provided by the local grid resource is evaluated first. Of course, if the requirement can be met locally, the job execution can be handled successfully without interaction with other external agents/nodes.

4. Performance Analysis

To show that the approach has the potential to become an acceptable distributed framework for self-management of computational resources in the next generation peer-to-peer grids, extensive simulation has been conducted upon four different Grid scenarios built in a random way upon some sample grid dimensional characteristics (small: 32 hosts and 512 tasks; average: 64 hosts and 1024 tasks; large: 128 hosts and 2048 tasks; and very large: 256 hosts and 4096 tasks). The grid networking topologies, modeled as non-oriented graphs, have been created using Waxman's method [15], [16]. In this method, the probability of the existence of link between two nodes *u* and *v* is given by:

$$P(u,v) = \varphi e^{-\frac{d}{\chi L}} \qquad (5)$$

where $0 < \varphi,\chi \leq 1$ are model parameters, d is the Euclidean distance between u and v and L is the maximum Euclidean distance between any two vertices of the graph. The available computational resources have been assigned randomly on all the grid nodes. In all the experiments, we used a dynamic model in which the job execution requests arrive at the grid according to a Poisson process with an arrival rate ε (jobs/second). The predicted job execution time is exponentially distributed with mean μ (1800 seconds in our tests).

4.1 Building the system model

The ACO meta-heuristic algorithm has been implemented in Java using the RePast multi-agent simulation framework. We used the Java version of Repast in order to take advantage of its great extensibility, ease of modifiability, portability, strict math and type definitions (to guarantee duplicatable results), and object serialization (to checkpoint out simulations). Repast is a free open source toolkit that was originally developed at the University of Chicago [17] and is now managed by the non-profit volunteer Repast Organization for Architecture and Development (ROAD). Repast seeks to support the development of extremely flexible models of living social agents, but is not limited to modeling living social entities alone. In short, our RePast simulation is primarily a collection of agents of both the search and resource management type and a model that sets up and controls the execution of these agents' behaviors according to a schedule. This schedule not only controls the execution of agent behaviors, but also actions within the model itself, determined by the ACO meta-heuristic paradigm.

4.2 Simulation results

All the results presented are taken from 100 iteration runs on 2 GHz HP Proliant DL380 machines running Linux, and each run was performed 10 times to collect the average makespan and flowtime values, that are the most interesting performance parameter for our evaluation. The proposed framework takes a comparatively long time to build solutions, approximately more then 10 seconds per iteration, so that the whole simulation took some hours. The ACO algorithm has been allowed to run for so long because this gives it reasonable time to build up a useful pheromone trail. The ants need some more running time to find solutions which significantly improve on the other solutions. The efficiency of the proposed solution can be easily observed from the graph in Fig. 3 below where the Makespan and Flowtime values measured as the results of the ACO meta-heuristic in our four typical Grid scenarios have been compared with the same values obtained by applying the classic Tabu Search (TS) approach as showed in [18].

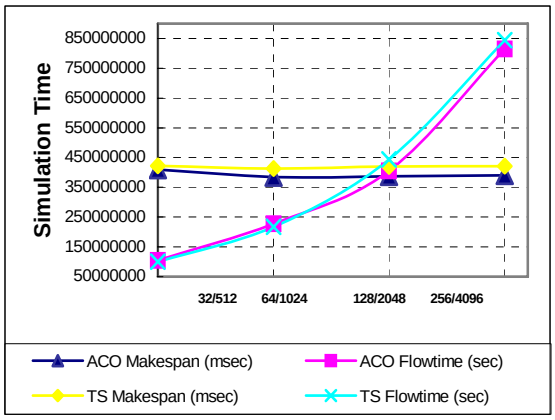

Figure 3. Makespan and flowtime performance

When representing both makespan and flowtime values simultaneously in the same graph we have to take into account that even though makespan and flowtime are measured in the same unit (seconds), the values they can take are in incomparable ranges, due to the fact that flowtime has a higher magnitude order over makespan, and its difference increases as more jobs and machines are considered. Consequently, in the above graph the makespan values have been scaled properly in msecs to be representable together with the flowtime ones. As can be seen, our ACO approach performs slightly better than the Tabu Search heuristic, especially in presence of larger problems, with more grid nodes and presented jobs, since in these cases the number of ants, or in other words the agents associated to tasks and nodes greatly increases. This result shows that the presented approach is a very good alternative for solving Job Scheduling in very large peer-to-peer Computational Grids where more priority is given to the reduction of Makespan than on immediate job scheduling time. The setting of the ACO parameters in our model will also affect the performance of the whole framework. Due to the time taken for a decent sized run of the ACO algorithm, and to the inbuilt stochasticity of the approach, finding the optimal values for these parameters has been a complex and very time-consuming task. For each topology used in our evaluation, the best observed values for the parameters for pheromone control α and β and ρ have been empirically determined through experiences on simulation results. At first, the pheromone evaporation parameter ρ, defining how quickly the ants "forget" past solutions has been always set to be in the range $0 \leq \rho \leq 1$. A higher value makes for a more aggressive search; in our tests a value of around 0.8 gave good results. The value of α determining the extent to which pheromone information is used as the ants build their solution, showed to be very critical for the success of the ACO search, and having tested values between 1-30, it seems that the ant-based algorithm works best with a relatively high value of

15 for all the topologies. Again, also for the parameter β, determining the extent to which heuristic information is used by the ants, all the values in the range 1-30 were tested, and a value near 10 worked well for all the tests and topologies. Once again, we observed that an higher β value may provide good solutions quickly, but a lower value may provide better results after a longer period of time. The best-performing values for the ACO model parameters used in our simulation are reported in the table 1 below.

Table 1. Best ACO parameters used in simulation

Parameter name	Parameter Value
α	15
β	10
ρ	0.8

The above values as experimentally determined in our simple tests work well enough, as the observed results show, but there is undoubtedly room for further improvement.

5. Related work and discussion

Job Scheduling on Computational Grids has taken considerable efforts from many researchers. There are several known meta-heuristics approaches, which explore the solution space and try to overcome local optimal solutions. Most of these approaches are based on a single heuristic method such as Genetic Algorithms (Braun et al. [19], Martino and Mililotti [20], Abraham et al. [21], Carretero and Xhafa [22]), Simulated Annealing (Yarkhan and Dongarra [23], Abraham et al. [21]), specifically focusing on Genetic Algorithms to achieve load balancing (Zomaya and Teh [36]) or using Tabu Search (Abraham et al. [21]). An hybrid approach based on an ACO algorithm combined with Tabu search is due to Ritchie and Levine [24][25]. Our approach is substantially different since it integrates the ACO technology with multi-agent systems, commonly recognized, because of their reactivity and their cooperation, negotiation, and competition capabilities as the best available solution to model highly dynamic and complex systems. What makes such a solution particularly attractive from the peer-to-peer Grid scheduling perspective is the fact that global properties like adaptation, self-organization and resilience are achieved without explicitly embedding them into the individual agents. In our model, there are no rules specific to initial conditions, unforeseen scenarios, and variations in the environment or presence of failures. Yet, given large enough agent/ant colonies, the global behavior is surprisingly adaptive and resilient. The agents' behavior, taken individually, may be easily understood, while the behavior of the scheduling system as a whole defies simple explanation. In other words, the interactions among agents, in spite of their simplicity, can give rise to richer and more complex patterns than those generated by single agents. Finally, while most of the scheduling schemes used in traditional peer-to-peer Grids, such as Sethi@home, are specialized in solving particular problems, our ACO-based framework aims at providing a general scheduling support for distributed computing, in which every independent node is capable of producing new jobs and introduce them in the network for computation.

6. Conclusions

The next generation grid computing environment must be intelligent and autonomous to meet requirements of smart self-management. Accordingly, we presented in this work a new adaptive strategy to efficiently distribute the jobs submitted on a grid on the available computational resources. The proposed approach is based on swarm intelligence and precisely on the ant colony optimization meta-heuristic implemented in a multi-agent system scenario. This fascinating family of algorithms tries to apply the ability of swarms to mathematical problems and was applied successfully to several optimization problems, so that they are widely recognized as one of the major self-organizing search mechanisms used in nowadays adaptive applications. One of the most interesting features of ant colony optimization-based approaches is that it may allow enhanced efficiency when the representation of the problem under investigation is spatially distributed and changing over time. On the other side, the Multi-Agent technology demonstrated to be an interesting solution to implement distributed and dynamic computational environments: agents confer the needed degree of autonomy to the system components and simplify the creation of dynamic relations among them. The agent-based ACO approach used in our work can be conceived as an initial attempt towards a distributed framework for building the next generation intelligent grid environments. We demonstrated that the above approach has at least two main advantages:

- first, the coordination and task distribution policies can rely on the interaction and self-organization capabilities of the ACO-based agents since they are high level system components which in a swarm-inspired organization naturally embed negotiation, competition and cooperation capabilities;
- also, the default services provided by multi-agent system meet typical peer-to-peer grid computing requirements; hence the use of a modular and extensible multi-agent system, simplifies and improve the efficiency in the Grid architecture development.

References

[1] T. De Fanti, C. De Laat, J. Mambretti, K. Neggers and B. St. Arnaud, "TransLight: A global-scale LambdaGrid for e-science", *Communications of the ACM,* 47(11), 2003.

[2] D. Anderson, "SETI@home", in A. Oram, editor, *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*, chapter 5. O'Reilly, 2001.

[3] The Anthrax Project, *http://www. chem. ox. uk/ anthrax.*

[4] Distributed.net Home Page, *http://www. distributed. net.*

[5] J. Kennedy, Y. Shi and R.C. Eberhart, "*Swarm Intelligence*", Morgan Kaufmann Publishers, San Francisco, 2001.

[6] M. Wooldridge, "*Intelligent Agents, in Multi-agent Systems – A Modern Approach to Distributed Artificial Intelligence*", G. Weiss Ed., Cambridge, MA, 1999.

[7] G. Weiss, "*Multi-agent Systems – A Modern Approach to Distributed Artificial Intelligence*", G. Weiss Ed., Cambridge, MA, 1999.

[8] M. Dorigo, G. Di Caro, "The Ant Colony Optimization Metaheuristic" in D. Corne, M. Dorigo and F. Glover (eds), *New Ideas in Optimization*, McGraw-Hill, 1999.

[9] M. Dorigo and L.M. Gambardella, "Ant colony System: Optimization by a colony of cooperating agents", *IEEE Transactions on Systems, Man, and Cybernetics* – Part B, vol. 26, no. 1, pp 29-41, 1996.

[10] J. Cao , D. P. Spooner , S. A. Jarvis , G. R. Nudd , "Grid load balancing using intelligent agents", *Future generation computer systems*, vol. 21, n. 1, pp. 135-149, Elsevier, 2005

[11] I. Foster, C. Kesselman, J. Nick, S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", Global Grid Forum, 2002.

[12] Xml protocol working group, *http://www.w3c.org/2000/xp/Group/.*

[13] W3C, WSDL specification, *http://www.w3c.org/TR/wsdl.*

[14] Uddi standards, *http://www.uddi.org*.

[15] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "*How to model an internetwork,*" in IEEE Infocom, vol. 2. San Francisco, CA: IEEE, pp. 594–602, 1996.

[16] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, 1988.

[17] E. Tatara, M.J. North, T.R. Howe, N.T. Collier, and J.R. Vos, "*An Introduction to Repast Modeling by Using a Simple Predator-Prey Example*", Proceedings of the Agent 2006 Conference, 2006

[18] T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen and R. F. Freund, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems", *Journal of Parallel and Distributed Computing*, 61(6):810–837, 2001.

[19] T.D. Braun, H.J. Siegel, N. Beck, L.L. Boloni, M. Maheswaran, A.I. Reuther, J.P. Robertson, M.D. Theys, and B. Yao, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems", *Journal of Parallel and Distributed Computing*, 61(6):810–837, 2001.

[20] V. Di Martino and M. Mililotti, "Sub optimal scheduling in a grid using genetic algorithms", *Parallel Computing*, 30:553–565, 2004.

[21] 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.

[22] J. Carretero and F. Xhafa, "*Using genetic algorithms for scheduling jobs in large scale grid applications*", in Workshop of the European Chapter on Metaheuristics EUME 2005, Metaheuristics and Large Scale Optimization. May 19-21, Vilnius, Lithuania, 2005.

[23] A. YarKhan and J. Dongarra, "*Experiments with scheduling using simulated annealing in a grid environment*", In GRID2002, pages 232–242, 2002.

[24] A.Y. Zomaya and Y.H. Teh, "Observations on using genetic algorithms for dynamic load-balancing", *IEEE Transactions On Parallel and Distributed Systems*, 12(9):899–911, 2001.

[25] G. Ritchie, "*Static multi-processor scheduling with ant colony optimisation & local search*", Master's thesis, School of Informatics, University of Edinburgh, 2003.

[26] G. Ritchie and J. Levine, "*A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments*", in 23rd Workshop PLANSIG 2004, 2004.