So-Grid: A Self-Organizing Grid Featuring Bio-Inspired Algorithms

AGOSTINO FORESTIERO, CARLO MASTROIANNI,
and GIANDOMENICO SPEZZANO
Institute for High Performance Computing and Networking (ICAR-CNR)

This article presents So-Grid, a set of bio-inspired algorithms tailored to the decentralized construction of a *Grid* information system that features adaptive and self-organization characteristics. Such algorithms exploit the properties of *swarm* systems, in which a number of entities/agents perform simple operations at the local level, but together engender an advanced form of *swarm intelligence* at the global level. In particular, So-Grid provides two main functionalities: logical reorganization of resources, inspired by the behavior of some species of ants and termites that move and collect items within their environment, and resource discovery, inspired by the mechanisms through which ants searching for food sources are able to follow the pheromone traces left by other ants. These functionalities are correlated, since an intelligent dissemination can facilitate discovery. In the Grid environment, a number of ant-like agents autonomously travel the Grid through P2P interconnections and use biased probability functions to: (i) replicate resource descriptors in order to favor resource discovery; (ii) collect resource descriptors with similar characteristics in nearby Grid hosts; (iii) foster the dissemination of descriptors corresponding to *fresh* (recently updated) resources and to resources having high quality of service (QoS) characteristics. Simulation analysis shows that the So-Grid replication algorithm is capable of reducing the entropy of the system and efficiently disseminating content. Moreover, as descriptors are progressively reorganized and replicated, the So-Grid discovery algorithm allows users to reach Grid hosts that store information about a larger number of useful resources in a shorter amount of time. The proposed approach features characteristics, including self-organization, scalability and adaptivity, which make it useful for a dynamic and partially unreliable distributed system.

Categories and Subject Descriptors: H.0 [**Information Systems**]: General; H.4 [**Information Systems Applications**]; I.2 [**Artificial Intelligence**]

General Terms: Algorithms

Additional Key Words and Phrases: Grid, multiagent systems, P2P, resource discovery, self-organization, swarm intelligence

This work has been partially supported by the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265) and by the SFIDA-PMI project cofunded by the Italian Ministry of Research and University, MIUR (reference number 4446/ICT).
Authors' email: {forestiero, mastroianni, spezzano}@icar.cnr.it.

ACM Reference Format:
Forestiero, A., Mastroianni, C., Spezzano, G. 2008. So-Grid: A self-organizing Grid featuring bio-inspired algorithms. ACM Trans. Autonom. Adapt. Syst. 3, 2, Article 5 (May 2008), 37 pages. DOI = 10.1145/1352789.1352790 http://doi.acm.org/10.1145/1352789.1352790

1. INTRODUCTION

Grid computing [Foster and Kesselman 2003] is an emerging computing model that provides the ability to perform higher throughput computing by taking advantage of many networked computers and distributing process execution across a parallel infrastructure. Modern Grids are based on the service oriented paradigm; for example, in the Globus Toolkit 4 based on the Web Services Resource Framework (WSRF [The Globus Alliance 2007]), resources are offered through the invocation of Web services, which boast enriched functionalities such as lifecycle and state management.

The *information system* is an important pillar of a Grid framework, since it provides information about Grid resources, critical to the operation of the Grid and the construction of applications. In particular, users turn to the information system to discover suitable resources that are needed to design and execute a distributed application, explore the properties of such resources and monitor their availability.

Owing to the dynamic nature of Grids, the set of available hosts and resources can change with time: hosts disconnect and join the network again, new resources and services may be added, existing ones can be removed, and the basic properties of a resource or a service may change. The dynamic nature of Grids makes human administrative intervention difficult or even unfeasible, thereby increasing the need for *self-organizing Grids* [Erdil et al. 2005], in which scalability is obtained through the use of automatic mechanisms and protocols.

The construction of self-organizing Grids can be favored by the use of bio-inspired algorithms, which are widely exploited to solve a number of complex problems (combinatorial algorithms, task allocation, routing problems, graph partitioning, etc.) [Bonabeau et al. 1999] and have been recently adopted to also provide distributed services in P2P networks [Babaoglu et al. 2002]. An interesting feature of many biological systems, ranging from ant colonies to wasp swarms and bird flocks, is the emergence of *swarm intelligence* despite the moderate complexity of individual components. In these systems, a number of small and autonomous entities perform very simple operations driven by local information (for example, while searching for food an ant follows a pheromone substance deposited by another ant, which has already discovered a food source; a bird adjusts its speed and direction by following the movements of nearby birds), but from the combination of such operations a complex and intelligent behavior emerges (ants are able to establish the shortest path towards a food source; birds travel in large flocks and rapidly adapt their movements to the ever changing characteristics of the environment) [Bonabeau et al. 1999; Dasgupta 2004].

Swarm biological systems can be quite naturally emulated in a distributed system through the multi-agent paradigm [Sycara 1998]: the behavior of

insects and birds can be imitated by mobile agents that travel through the hosts of a Grid and perform simple operations. Agent-based systems may inherit useful and beneficial properties from biological counterparts, namely: (i) *self-organization*, since decisions are based on local information without any central coordinator; (ii) *adaptivity*, since agents can react flexibly to the ever-changing environment; (iii) *stigmergy awareness* [Grasse' 1959], since agents are able to interact and cooperate through the modifications of the environment that are induced by their operations.

This article presents *So-Grid*, a bio-inspired approach for the construction of a self-organizing information system, which allows the efficient management and discovery of resources in a dynamic Grid. The main purpose of this approach is the replication and dissemination of metadata documents that provide basic information about Grid resources, in order to advertise their presence and foster their use. In particular, a metadata document (in the following also called *descriptor* for the sake of simplicity) can be composed of a syntactical description of the service (a WSDL—Web Services Description Language—document) and/or an ontology description of service capabilities. This objective is typical of epidemic mechanisms, which disseminate information in distributed systems [Petersen et al. 1997]. However, in So-Grid, descriptors are not only replicated but also reorganized according to their features. More specifically, descriptors of similar resources—of resources belonging to the same *class*—are accumulated in restricted regions of the Grid. This reorganization not only allows information management to be simplified, since neighbor hosts manage homogeneous information, but allows the definition of a discovery algorithm that can direct user queries towards a Grid region where a number of useful descriptors can be discovered in a short amount of time.

The two most important So-Grid algorithms are: the *replication algorithm*, which performs replication and reorganization of descriptors, and the *discovery algorithm*, which drives user queries towards descriptors. These two algorithms were presented, in their basic versions, respectively in Forestiero et al. [2005] and Forestiero et al. [2006]. Here we better specify and tune their features; introduce their enhanced versions, through which it is possible to foster the dissemination of information pertaining to high quality and/or recently updated resources; and present a large set of new results obtained with an event-based simulation framework, which is now also available online.

The *replication algorithm* is inspired by the behavior of ants and termites, which cluster and map corpses [Deneubourg et al. 1990]. A number of replication agents travel the Grid through P2P interconnections among Grid hosts, and *pick* and *drop* resource descriptors according to appropriate probability functions. The objective of agents is twofold: (i) the replication of descriptors and (ii) their spatial reorganization. The two objectives are achieved by the definition of two operating *modes* of agents. Agents are generated by new or reconnecting hosts, and initially work in the *copy* mode: they can replicate descriptors and disseminate them on the Grid. However, when an agent realizes, from its own past activities, that the generation of other replicas would spoil the reorganization of descriptors, it switches to the *move* mode: it will only move descriptors from one host to another without generating new replicas. Interestingly, the

mode switch is performed autonomously, only on the basis of local information and a pheromone mechanism. This is essential to preserve the self-organizing nature of the algorithm and assure a scalable behavior.

The *discovery algorithm* is devised to exploit the logical resource organization achieved by the replication algorithm. The rationale is the following: if a large number of descriptors of a specific class are accumulated in a restricted region of the Grid, it becomes convenient to drive search requests (issued by users to search for resources of that class) towards that region, in order to maximize the number of discovered resources and minimize the response time. Therefore, Grid hosts that accumulate a large number of descriptors of a given class are elected as *representative peers* and are used as attractors for discovery requests. A discovery operation is performed in two phases. In the first phase, the *random walk* technique is adopted [Lv et al. 2002]: a number of asynchronous query messages are issued by the requesting peer and will travel the Grid in parallel in a *blind* fashion. In the second phase, whenever a query message gets close enough to a representative peer, the search becomes *informed*: the query message is driven towards the representative peer and will likely discover a large number of useful descriptors. The discovery algorithm is referred to as *semi-informed*, since it combines the benefits of both *blind* and *informed* resource discovery approaches, which are currently used in P2P networks [Tsoumakos and Roussopoulos 2003b].

These two algorithms are executed concurrently and continuously by different types of agents, thus achieving a fruitful form of division of labor: while *replication* agents replicate and reorganize information, *discovery* agents exploit the work of the former to find useful information more rapidly and effectively. The decentralized and self-organizing nature of these algorithms allow them to respond rapidly to system changes, for example to disconnections and reconnections of hosts and to modifications of resource characteristics. Moreover, So-Grid is intrinsically scalable, since agent operations are only based on local information, and do not rely on any central entity that could constitute a bottleneck, especially in very large systems.

The So-Grid replication algorithm is also available in an enhanced version that can be adopted if the aim is to improve not only the quantity, but also the quality and freshness of resources that can be discovered by users. The adopted strategy is to foster the dissemination of descriptors corresponding to high QoS and/or recently updated resources. This will obviously increase the probability of discovering such descriptors on the Grid, thus improving the level of satisfaction of users.

Actually, the selective dissemination of descriptors of high QoS resources is possible only if there is a commonly accepted interpretation of high quality, and if an ordering is defined among resources, through which it is possible to determine, between two resources, which is better. For example, the QoS can be the average response time of a software, the computing power of a cluster, the cost of a service, or simply the ranking value given by users to a homogeneous set of resources/services. On the other hand, if the QoS is defined in a more complex way and, more importantly, if it is based on the subjective perception of users, the advanced version of the So-Grid replication algorithm is not exploitable.

Conversely, it is always possible to foster the dissemination of fresh resources, as the age of a resource can be defined as the amount of time elapsed since the last time the resource was modified. This is an important feature in Grids, since the discovery of dynamic resources (e.g., amount of free main memory in a host) is generally more critical than discovery of static resources (e.g., read-only documents) [Cheema et al. 2005].

A simulation analysis was carried out to demonstrate the effectiveness of the So-Grid algorithms, their ability to disseminate, reorganize and discovery resource descriptors, and their good scalability properties, which derive from the decentralized and self-organizing characteristics of the biological systems by which they are inspired. Simulation results also show that it is possible to tune the replication algorithm in order to enforce or reduce the selective dissemination of high quality and/or fresh resource descriptors. Simulations can be performed by the reader with an event-driven simulator made available through the So-Grid Portal, at the Web address *http://so-grid.icar.cnr.it*.

The remainder of the article is organized as follows: Section 2 introduces the So-Grid replication algorithm, both in its basic and its enhanced version; Section 3 analyzes the performance of the replication algorithm, and evaluates its scalability and adaptivity properties; Section 4 introduces and examines the So-Grid discovery algorithm, whose performance is evaluated in Section 5; related work is discussed in Section 6, and Section 7 concludes the article.

2. THE SO-GRID REPLICATION ALGORITHM

The main purpose of the replication algorithm is to achieve a logical organization of Grid resources by spatially sorting resource descriptors over the Grid according to their classifications.

It is assumed that the resources are classified into a number of classes N_c, according to their semantics and functionalities. The rationale of this classification is that users usually issue a query not to search for a single specific resource, but to collect information about resources having specified characteristics [Crespo and Garcia-Molina 2002], for example a host with a given CPU power or a bioinformatic software able to perform particular operations on protein data. A *class of resources* is therefore defined as a set of Grid services/resources having specified properties. After issuing a query, a user can discover a number of resources of a given class, and then can choose the resources that are the most appropriate for their purposes.

It is also assumed that the Grid system uses P2P interconnections to enable communications and document exchanges among Grid hosts. This is coherent with the recent effort to adopt P2P techniques in Grid frameworks, in order to enhance efficiency and scalability features of large-scale Grids [Iamnitchi et al. 2003].

In Section 2.1, we discuss the basic pick and drop operations that are performed by agents. Then, in Section 2.2, we describe how So-Grid handles the dynamic nature of the Grid and manages the turnover of agents. Section 2.3 presents the pheromone mechanism through which an agent self-determines its operating mode. Finally, Section 2.4 describes the enhanced versions of the

pick and drop probability functions that enable the selective dissemination of information pertaining to high quality and fresh resources.

2.1 Basic Pick and Drop Operations

Each replication agent offers its contribution to the reorganization of descriptors. When a peer connects to the network, it may generate an agent that sets off from this peer and performs a number of hops through the P2P links that interconnect the Grid hosts. Agents perform *pick* and *drop* operations to replicate and move descriptors from one peer to another. When arriving at a host, each agent autonomously decides whether or not to *pick* the descriptors of a given class and then carry them in its successive movements, or to *drop* descriptors that it has previously picked from another host. Each host can store descriptors of local resources, which are never removed, as well as descriptors of resources published by other hosts, which can be picked up and discarded. When distinction is relevant, such descriptors will respectively be referred to as *local* and *remote* descriptors.

The *pick* and *drop* operations are driven by the corresponding probability functions that are defined and discussed in Sections 2.1.1 and 2.1.2. These functions are inspired by the mechanisms introduced in Deneubourg et al. [1990], and later elaborated and discussed in Bonabeau et al. [1999] and Martin et al. [2002], to emulate the behavior of some species of ants that build cemeteries by aggregating corpses in clusters.

However, these mechanisms are adapted to our purposes, by making the following main modifications: (i) descriptors are not only aggregated, as in the mentioned papers, but also *replicated*, and two different ant modes are defined to balance these two functionalities; (ii) descriptors are spatially mapped: accumulated in different clusters according to the class to which they belong.

A high-level description of the replication algorithm, performed by replication agents, is given in the flowchart of Figure 1. Periodically, each agent performs a small number of P2P hops among Grid hosts. Whenever an agent arrives at a new Grid host, for every resource class, it evaluates the pick or drop probability function, specifically: (i) if the agent does not carry any descriptor of this class, it evaluates the *pick probability function*, so as to decide whether or not to pick the descriptors of this class from the current host; (ii) if the agent already carries some descriptors of this class, it evaluates the *drop probability function*, so as to decide whether or not to leave these descriptors in the current host. After picking the descriptors of a class, the agent will carry them until it drops them into another host, and then will try to pick other descriptors from another host.

The *pick* operation can be performed with two different modes. If the *copy* mode is used, the agent, when executing a pick operation, leaves the descriptors on the current host, *generates a replica* of them, and carries these descriptors until it drops them in another host. Conversely, with the *move* mode, as an agent picks the descriptors, it removes them from the current host (except for the *local* descriptors, which cannot be removed), thus preventing an excessive proliferation of replicas. The use of these two modes is discussed in

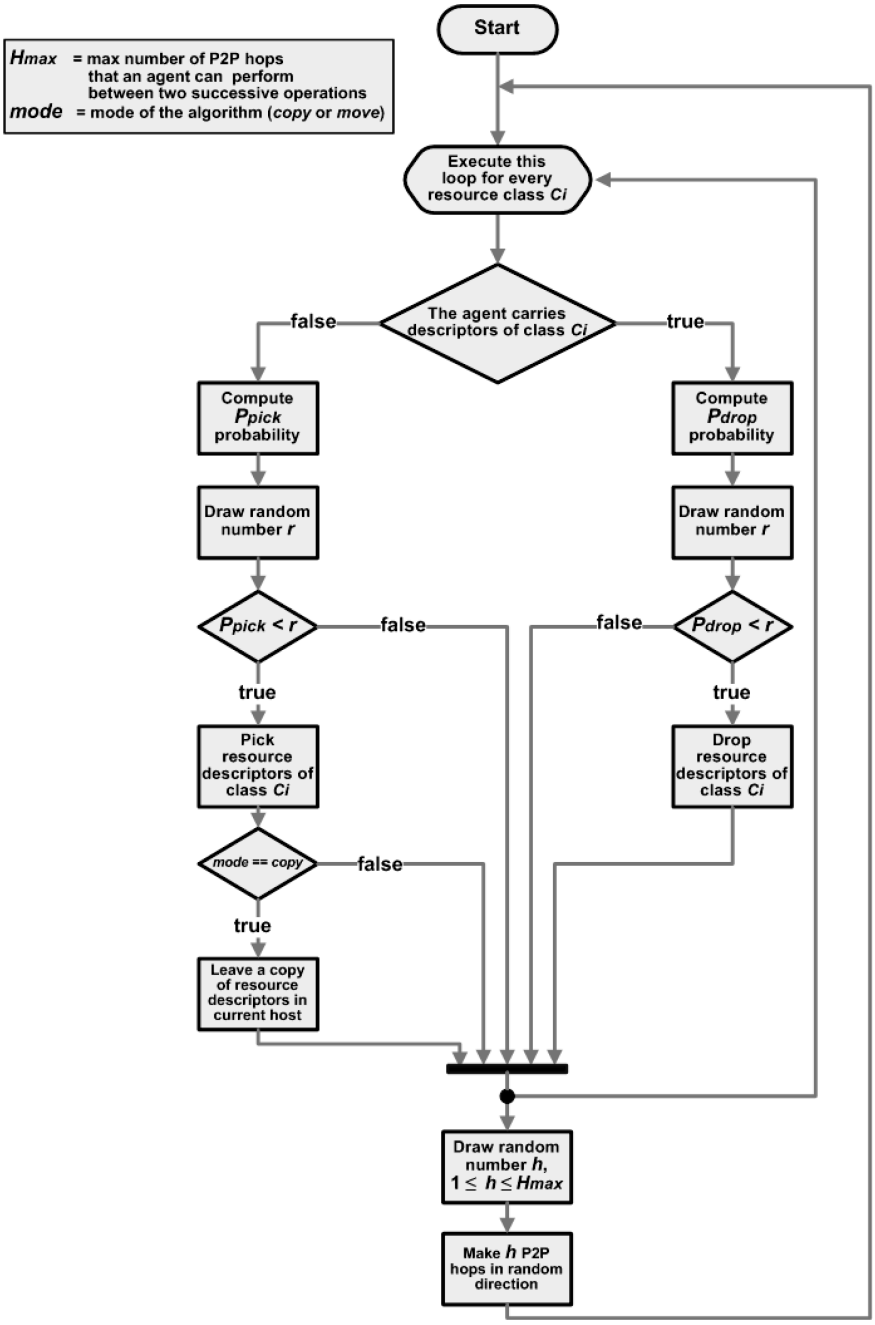

Fig. 1. The algorithm performed by So-Grid replication agents.

Section 2.3. The following two subsections discuss the probability functions that drive agent operations.

2.1.1 *Basic Pick Probability Function.* The probability of picking the descriptors of a given class must decrease as the local region of the Grid accumulates these descriptors and vice versa. This assures that as soon as the equilibrium condition is broken (i.e., descriptors belonging to different classes begin to be accumulated in different regions), a further reorganization of descriptors is increasingly driven. The basic P_{pick} probability function, defined in Formula (1), is the product of two factors, which take into account, respectively, the absolute accumulation of descriptors of a given class and their relative accumulation with respect to other classes. While the *absolute* factor is inspired by the pick probability defined in Bonabeau et al. [1999], the *relative* factor was appositely introduced to achieve the spatial separation of descriptors belonging to different classes.

$$P_{pick} = \left(\frac{fa^2}{k1 + fa^2}\right)^2 \cdot \left(\frac{k2}{k2 + fr}\right)^2. \tag{1}$$

The *fa* fraction is computed as the number of *local* descriptors of the class of interest, stored by the hosts located in the *visibility region*, out of the overall number of descriptors of the class of interest, both *local* and *remote*, that are stored by the same hosts. The value of *fa* is comprised between 0 and 1, as well as the value of *fr*, defined in the following. The *visibility region* includes all the hosts that are reachable from the current host with a given number of hops within the *visibility radius*, which is an algorithm parameter. As more remote descriptors of a class are accumulated in the local region, *fa* decreases, the first factor of the pick function decreases as well, and vice versa, which is the desired behavior.

Conversely, the *fr* fraction is computed as the number of descriptors of the class of interest, accumulated in the hosts located in the visibility region, divided by the overall number of descriptors of all classes that are accumulated in the same region. As the local region accumulates more descriptors of a class, with respect to other classes, *fr* increases, the value of the pick probability for this class becomes lower, and vice versa. *k1* and *k2* are non-negative constants, both set to 0.1, as in Bonabeau et al. [1999].

2.1.2 *Basic Drop Probability Function.* Whenever an agent gets to a new Grid host, if it is carrying descriptors of a given class, it must decide, whether or not to drop these descriptors in the current host. Like the pick function, the drop function is first used to break the initial equilibrium and then to strengthen the spatial mapping of descriptors. However, as opposed to the pick probability, the drop probability for a class, shown in Formula (2), increases as the local region accumulates descriptors of this class. In (2), the *fr* fraction is defined as in formula (1), whereas the threshold constant *k3* is set to 0.3 [Bonabeau et al. 1999].

$$P_{drop} = \left(\frac{fr}{k3 + fr}\right)^2. \tag{2}$$

Notice that the drop function depends only on the relative accumulation of descriptors, and it does not contain any factor related to absolute accumulation. Indeed it has been observed that in the long term an "absolute" factor could start an avalanche mechanism, which occurs when a region accumulates many descriptors belonging to two or more different classes. In this case, drop operations would be further facilitated for all such classes, instead of only one, thus weakening the spatial reorganization of descriptors. This phenomenon does not occur with the mere use of a relative accumulation factor.

2.2 Handling a Dynamic Grid

In a dynamic Grid, peers can go down and then reconnect, and resources can change their characteristics. To account for this, two characteristics are taken into consideration: the *average connection time* of a peer and the *update interval* of resources.

The average connection time of a specific peer is generated according to a Gamma probability function, with an average value set to the parameter T_{peer}. Use of the Gamma distribution assures that the Grid contains very dynamic hosts, which frequently disconnect and rejoin the network, as well as much more stable hosts. The rate at which resources are updated also has a significant impact on the availability of dynamic resources, as discussed in Cheema et al. [2005]. In the present work it is assumed that the characteristics of a resource are updated after an average time interval equal to $T_{resource}$. This parameter assumes different values for different peers (according to a Gamma distribution), thus assuring the presence of both nearly static and highly dynamic resources.

As a consequence of this dynamic nature, two issues must be tackled. The first is related to the management of new resources provided by new or reconnected hosts. Indeed, if all the replication agents switch to the *move* mode, it becomes impossible to replicate and disseminate descriptors of new resources; as a consequence, agents cannot be allowed to live forever, and must gradually be replaced by new agents that set off in the *copy* mode. The second issue is that the system must (i) remove *ghost descriptors*, descriptors of resources provided by hosts that have left the system, and therefore are no longer available, and (ii) limit the presence of *obsolete descriptors*, which have obsolete information about resources, because these resources have changed their characteristics.

Simple mechanisms are adopted to cope with these two issues. The first is to correlate the lifecycle of agents to the lifecycle of peers. When joining the Grid, a host generates a number of agents given by a discrete Gamma stochastic function, with average N_{gen}, and sets the lifetime of these new agents to the average connection time of the peer itself. This setting assures that (i) a proper turnover of agents is achieved, because old agents die when their lifetimes expire and new agents are generated by reconnecting peers, and (ii) the relation between the number of peers and the number of agents is maintained with time (more specifically, the overall number of agents is on average equal to the number of active peers multiplied by N_{gen}, as confirmed by simulation tests). The agent turnover allows for the dissemination of descriptors of new resources, since new agents start in the *copy* mode.

A second mechanism assures that every time a peer disconnects from the Grid, it loses all the descriptors previously deposited by agents, thus contributing to the removal of obsolete descriptors. Finally, a *soft state* mechanism [Sharma et al. 1997] is adopted to avoid the accumulation of obsolete descriptors in very stable nodes. Each host periodically refreshes the descriptors of the resources owned by other hosts, by contacting those hosts and retrieving from them updated information about the resources.

It is worth mentioning that this approach for handling a dynamic Grid implicitly manages any unexpected peer fault, because this occurrence is processed in exactly the same way as a peer disconnection. Indeed, the two events are indistinguishable, since (i) a peer does not have to perform any procedure before leaving the system, and (ii) in both cases, disconnection or fault, the remote descriptors that the peer has accumulated are removed from the system.

2.3 Self-Tuning of Agents

The effectiveness of the replication algorithm is evaluated through a spatial entropy function, which is based on the well-known Shannon's formula for the calculation of information content. For each peer p, the local entropy E_p, defined in formula (3), gives an estimation of the extent to which the descriptors have been spatially mapped within the visibility region centered in p. In (3), *fr(i)* is the fraction of descriptors of class C_i that are located in the visibility region with respect to the overall number of descriptors located in the same region. The E_p function is normalized (the number of classes is equal to N_c), so that its value is between 0 and 1. In particular, an entropy value equal to 1 corresponds to the presence of comparable numbers of descriptors belonging to all the different classes, whereas a low entropy value is obtained when the region centered in p has accumulated a large number of descriptors belonging to one specific class, thus contributing to the spatial mapping of descriptors. As shown in Formula (4), the overall entropy, E, is defined as the average of the entropy values, E_p, computed at all the Grid peers (the number of peers is equal to N_p).

$$E_p = \frac{\sum_{i=1\ldots N_c} fr(i) \cdot \lg \frac{1}{fr(i)}}{\lg N_c}. \tag{3}$$

$$E = \frac{\sum_{p \epsilon Grid} E_p}{N_p}. \tag{4}$$

The overall spatial entropy can be minimized if agents work under both their operating modes: *copy* and *move*. In the first phase of its life, each agent must *copy* the descriptors that it picks from a Grid host, but when it realizes from its own activeness that the mapping process is at an advanced stage, it must only *move* descriptors from one host to another, without creating new replicas. In fact, the *copy* mode cannot be maintained for a long time, since eventually every host would have a very large number of descriptors of all classes, thus weakening the efficacy of spatial mapping.

An approach based on ants' pheromone [Van Dyke Parunak et al. 2005] enables each agent to perform this mode switch, from *copy* to *move*, only on the basis of local information. This approach is inspired by the observation that

agents perform more operations when the system entropy is high, but operation frequency gradually decreases as descriptors are properly reorganized. The reason for this is that the values of P_{pick} and P_{drop} functions, defined in formulas (1) and (2), decrease as descriptors are reorganized on the Grid. Each agent maintains a pheromone base (a real value) and increases it when its activeness tends to decrease; the agent switches to the *move* mode as soon as the pheromone level exceeds a defined threshold, H_r. In particular, at given time intervals, every 2000 seconds,[1] each agent counts the number of times that it has evaluated the pick and drop probability functions, $N_{attempts}$, and the number of times that it has actually performed pick and drop operations, $N_{operations}$. At the end of each time interval, the agent makes a deposit, which is inversely proportional to the fraction of performed operations, into its pheromone base. An evaporation mechanism is used to give a greater weight to the recent behavior of the agent. At the end of the i-th time interval, the pheromone level Φ_i is computed with formulas (5) and (6).

$$\Phi_i = Ev \cdot \Phi_{i-1} + \phi_i. \tag{5}$$

$$\phi_i = 1 - \frac{N_{operations}}{N_{attempts}}. \tag{6}$$

The evaporation rate Ev is set to 0.9 [Van Dyke Parunak et al. 2005], and ϕ_i is the amount of pheromone deposited in the last time interval. The pheromone level can assume values between 0 and 10: the upper limit can be obtained by equalizing Φ_i to Φ_{i-1} and setting ϕ_i to 1. As soon as the pheromone level exceeds the threshold H_r (whose value must also be set between 0 and 10), the agent switches its protocol mode from *copy* to *move*. The value of H_r can be used to tune the number of agents that work in the *copy* mode and therefore are able to create new resource descriptors. The effect of this tuning is discussed in Section 3.2.

This pheromone mechanism is a form of *sematectonic* stigmergy [Camazine et al. 2001], since agents communicate with each other indirectly and base their actions on the current state of the environment, specifically on the quantity and type of descriptors present in the local Grid region.

2.4 Enhanced Pick and Drop Probability Functions

The pick and drop functions discussed in Sections 2.1.1 and 2.1.2 allow resource descriptors to be replicated and reorganized regardless of their quality of service and their dynamic properties. Better results can be obtained if descriptors of resources having high QoS characteristics are replicated and disseminated more rapidly and effectively than descriptors of low QoS resources. This way a user who issues a query will likely find more valuable resources

[1]The choice of updating the pheromone level at every time interval, instead of at every single agent operation, was made to fuse multiple observations into a single variable, so giving a higher statistical relevance to the decisions of the agent. The 2000-seconds value allows on average 33.3 operations to be aggregated, since the average interval between two agent movements is set to 60 seconds.

and his/her satisfaction will increase [Ran 2003; Vu et al. 2005]. Analogously, descriptors of dynamic resources should be disseminated more rapidly than those of static resources, in order to provide timely information to Grid users [Cheema et al. 2005]. These two issues are tackled by the So-Grid replication algorithm through the definition of *enhanced* pick and drop functions.

It should be remarked, as discussed in the introductory section, that these functions are exploitable only if there is a general agreement about the definition of high quality and if resources can be ordered with respect to their quality. In such cases, it can be assumed that the QoS of a resource is expressed by a non-negative real value ranging from 0 to 10, with higher values corresponding to better quality (QoS values are assumed to be uniformly distributed, with average 5.0). The QoS value of a resource is inserted in a specific field of the corresponding descriptor; therefore, in the following, we will not distinguish between the QoS level of a *descriptor* and the QoS level of the corresponding *resource*.

To achieve a selective dissemination that favors the propagation of high QoS descriptors, pick and drop functions are enhanced through the definition of additional factors. More specifically, the pick probability defined in formula (1) is multiplied by the factor $FQoS_{pick}$ defined in formula (7).

$$FQoS_{pick} = \frac{k4}{k4 + \frac{QoS_{descriptor} - QoS_{peer}}{QoS_{descriptor}}}. \tag{7}$$

This factor is evaluated by a replication agent each time it moves to a new host. In Formula (7), $QoS_{descriptor}$ is the average QoS value of a generic descriptor. Each agent estimates $QoS_{descriptor}$ by averaging the QoS values of the descriptors stored by the peers that it visits. QoS_{peer} is the average QoS value of the descriptors of the class of interest that are stored by the current peer. Parameter *k4* is set to a real value not lower than 2. It can be noticed that the average value of $FQoS_{pick}$ is equal to 1, which assures that the overall replication and mapping of descriptors (without considering their QoS values) is not biased by the new factor. However, if the current peer stored descriptors of the class of interest characterized by high QoS, the value of $FQoS_{pick}$ is higher than 1; hence the overall pick probability increases,[2] and the pick operation is favored. Conversely, the value of $FQoS_{pick}$ is lower than 1 if the current peer stored low QoS descriptors, which is the desired behavior.

Analogously, the drop probability function defined in Formula (2) is multiplied by a new factor $FQoS_{drop}$, reported in formula (8).

$$FQoS_{drop} = \frac{k4}{k4 + \frac{QoS_{peer} - QoS_{agent}}{QoS_{agent}}}. \tag{8}$$

[2]With the use of the new factor, in some, though very rare, cases the pick probability can assume values higher than 1; in these cases the probability is truncated to 1. It corresponds to a 100% probability of picking descriptors that have a very high QoS. Analogous considerations hold for the factors defined in Formulas (8) and (9).

In this formula, QoS_{agent} is the average QoS value of the descriptors of the class of interest that are carried by the agent. The drop operation is favored when the agent carries descriptors that have QoS values higher than the descriptors located in the current peer, thus fostering the dissemination of high quality descriptors.

In a similar fashion, the replication algorithm is enhanced to foster the dissemination of information about recently updated resources. To achieve this, the pick probability function defined in Formula (1), is multiplied by the factor $FAge_{pick}$, defined in Formula (9). In this factor, $AGE_{resource}$ is the average *age of resources* present in the Grid. The *age of a resource* is defined as the amount of time elapsed since the last time that the resource was modified. As mentioned in Section 2.2, resource characteristics are updated with an average time interval time equal to $T_{resource}$. The parameter $AGE_{resource}$ is estimated by an agent by averaging the resource age of all the resources stored by the peers that it visits. On the other hand, the parameter AGE_{peer} is defined as the average age of the resources stored by the peer currently visited by the agent. The parameter $k5$ is set to a real value not lower than 2.

$$FAge_{pick} = \frac{k5}{k5 + \frac{AGE_{peer} - AGE_{resource}}{AGE_{resource}}}. \tag{9}$$

As for the $FQoS_{pick}$ factor, it can be observed that the average value of the factor $FAge_{pick}$ is equal to 1, which assures that the overall replication of descriptors (without considering the dynamic nature of corresponding resources) is not biased by the new factor. However, the new factor fosters the replication of descriptors of recently updated resources, which most likely are also the resources characterized by a higher dynamic nature. In fact, if $AGE_{peer} < AGE_{resource}$, the factor $FAge_{pick}$ becomes higher than 1, so the overall pick probability increases. It was not found necessary to include a new factor in the drop probability function. When a replication agent picks descriptors related to recently updated resources, such descriptors are spontaneously disseminated through subsequent drop operations.

In conclusion, the enhanced pick and drop probability functions are defined, respectively, by Formulas (10) and (11). In Section 3.4, it will be observed that the overall performance of the replication algorithm is not affected by use of the additional factors (denoted as optional), but these factors can be exploited to foster the selective dissemination of descriptors corresponding to high QoS and/or recently updated resources.

$$P_{pick}(enhanced) = P_{pick} \cdot [FQoS_{pick}] \cdot [FAge_{pick}]. \tag{10}$$

$$P_{drop}(enhanced) = P_{drop} \cdot [FQoS_{drop}]. \tag{11}$$

3. PERFORMANCE OF THE REPLICATION ALGORITHM

The performance of the So-Grid algorithms was evaluated with an event-based simulator written in Java. Simulation objects are used to emulate Grid peers and So-Grid agents. Each object reacts to external events according to a

finite state automaton and responds by performing specific operations and/or by generating new messages/events that are delivered to other objects.

For example, a peer visited by a replication agent gives it information about the resources and descriptors that this peer stores; afterwards the agent uses probability functions to decide whether or not to pick descriptors from, or drop descriptors into, the peer. Finally, the agent sets the simulation time in which it will perform its next movement on the Grid and creates a related event that will be delivered to the destination peer at the specified time. Events are ordered in a common queue and are delivered to corresponding destination objects according to their expiration time, so that peers and agents can operate concurrently along the simulation time.

Simulations can be performed by the reader with an event-driven simulator made available through the So-Grid Portal, at the Web address *http://so-grid.icar.cnr.it*. The portal allows the user, after a simple registration procedure, to set a large number of network and algorithm parameters, run simulations, graphically monitor performance indices at run time, download results, and store them on a personalized storage space on the server. Moreover, the user can graphically compare the results obtained with simulation tests made in the same or in different sessions, which enables an interesting parameter sweep analysis.

3.1 Simulation Scenario and Performance Indices

Grid networks having a number of hosts N_p ranging from 1000 to 7000 peers are considered in this work, but the default value is set to 2500. Hosts are linked through P2P interconnections, and each host is connected to four peer hosts on average. The topology of the network is built by using the well-known scale-free algorithm defined by Barabási and Albert [1999], that incorporates the characteristic of preferential attachment (the more connected a node is, the more likely it is to receive new links) that was proved to exist widely in real networks. The number of Grid resources owned and published by a single peer is obtained with a Gamma stochastic function with an average value equal to 15 (see Iamnitchi et al. [2003]). Grid resources are assumed to be classified in a number of classes N_c, set to 5 by default, but varied from 5 to 30 in specific tests.

The average connection time of a peer, T_{peer} (see Section 2.2), is varied from 35000 to 1000000 seconds (with the default value set to 100000 seconds), whereas the update interval of resources, $T_{resource}$, is set to 50000 seconds. The mean number of agents generated by a single peer, N_{gen}, is set to 0.5; as a consequence, the average number of replication agents, N_a, that travel the Grid is $N_p/2$, as explained in Section 2.2. The average time, T_{mov}, between two successive agent movements (between two successive evaluations of pick and drop functions) is 60 s, whereas the maximum number of P2P hops that are performed within a single agent movement, H_{max}, is set to 3, in order to limit the traffic generated by agents. The visibility radius, R_v, used for the evaluation of pick and drop functions (see Section 2.1), is set to 1, which means that these functions are based exclusively on very local information. Finally, the

Table I. Simulation Parameters

Parameter	Value
Grid size (number of peers), N_p	1000 to 7000
Average number of neighbor peers	4 (power law network)
Mean number of resources published by a peer	15
Number of classes of resources, N_c	5 to 30
Mean peer connection time, T_{peer}	35000 s to 1000000 s
Mean update interval of resources, $T_{resource}$	50000 s
Number of agents, N_a	$N_p/2$
Mean time between two successive movements of an agent, T_{mov}	60 s
Maximum number of hops, H_{max}	3
Visibility radius, R_v	1
Pheromone threshold, H_r	3 to 10

pheromone threshold H_r, defined in Section 2.3, is set to values ranging from 3 to 10.

It is worth noting that the So-Grid replication algorithm is very robust with respect to variations of these parameters. For example, the average number of resources published by a peer, the number of resource classes, the average connection time of a peer, and so on, can affect the rapidity of the process, and in some cases can slightly influence the steady values of performance indices, but the qualitative behavior is always preserved.

A set of performance indices is defined for the performance evaluation of the replication algorithm. The overall entropy, E, defined in Section 2.3, is used to estimate the effectiveness of the algorithm in the reorganization of descriptors. The N_{dpr} index is defined as the mean number of descriptors, both local and remote, that are generated for each resource. Since new descriptors are only generated by replication agents that work in the *copy* mode, the number of such agents, N_{copy}, is another interesting performance index. The processing load, L, is defined as the number of agents per second that get to a single peer, and there evaluate pick or drop operations.

Finally, two more performance indices are defined to evaluate the effectiveness of the enhanced version of the algorithm, which uses enhanced pick and drop functions (Section 2.4). $QoS_{descriptor}$ is the average QoS value of resource descriptors, both local and remote, that are present in the Grid at a given time. $AGE_{descriptor}$ is the average *age of a descriptor*, defined as the amount of time elapsed since the last time that the related resource was modified. Since a remote descriptor cannot be directly informed of the updates performed on the corresponding resource, it often happens that the age of a descriptor is higher than the age of the corresponding resource. This occurs when a resource is updated after the descriptor has been picked and moved away by an agent.

A graphical description of the behavior of the replication algorithm is given in Figure 2. For the sake of clarity, here the number of classes N_c is set to 3, peers are connected in a bidimensional mesh instead of a scale-free topology, and only a portion of the Grid is shown, though the simulation was performed on a network with 2500 hosts. Different symbols and colors are associated with the three classes. Each peer is marked with the symbol and color that corresponds to the class to which the largest number of descriptors, stored by this

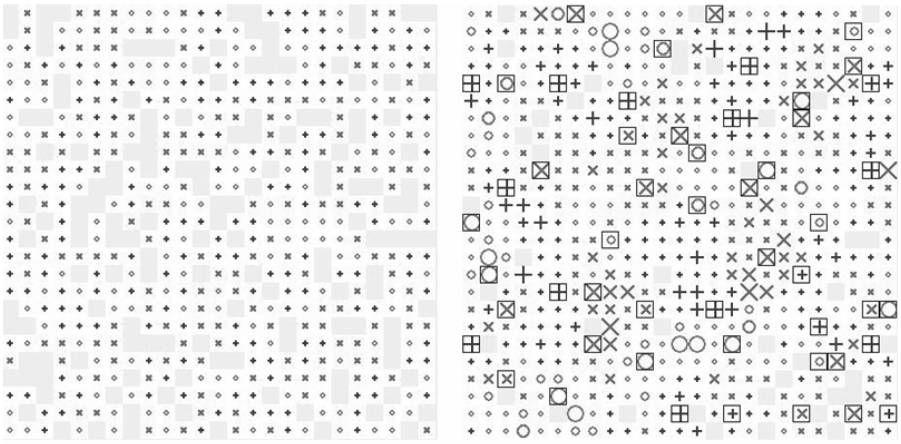

Fig. 2. Accumulation and reorganization of resource descriptors, belonging to 3 resource classes, in a region of the Grid.

peer, belong. Furthermore, the symbol size is proportional to the number of descriptors of the dominant class. For example, if circles correspond to class C_1, a peer marked with a big circle stores many descriptors of this class, while a peer marked with a smaller circle stores fewer descriptors of the same class. In both cases, the descriptors of class C_1 are the most numerous in this peer. The peers with a thicker border are *representative peers*, which work as attractors for query messages issued to discover resources of the dominant class, as described in Sections 4.1 and 4.2. Two snapshots of the network are shown: the first is taken when the replication process is initiated (time 0), while the second is taken 400000 seconds later, in a quite steady situation. This figure shows that descriptors are initially distributed in a completely random fashion, but subsequently they are accumulated and reorganized by replication agents in separate regions of the Grid, according to their class.

3.2 Tuning of the Replication Algorithm

The So-Grid replication algorithm was evaluated both in its basic version, which uses basic pick and drop functions (see Section 2.1) and in its enhanced version, which exploits enhanced pick and drop functions (see Section 2.4). A first set of simulation tests was performed to evaluate the basic replication algorithm. The number of classes N_c was set to 5 and the Grid size to 2500.

The first objective was to verify the effectiveness of the stigmergy mechanism, which drives the mode switch of agents. Figure 3 reports N_{copy}, the number of agents that work in *copy* mode (also called *copy agents* in the following) versus time, for different values of the pheromone threshold H_r. When the process is initiated, all the agents (about 1250, half the number of peers) are generated in the *copy* mode, but subsequently several agents switch to *move*, as soon as their pheromone value exceeds the threshold H_r. This corresponds to the sudden drop of curves that can be observed in Figure 3. This drop does not occur if H_r

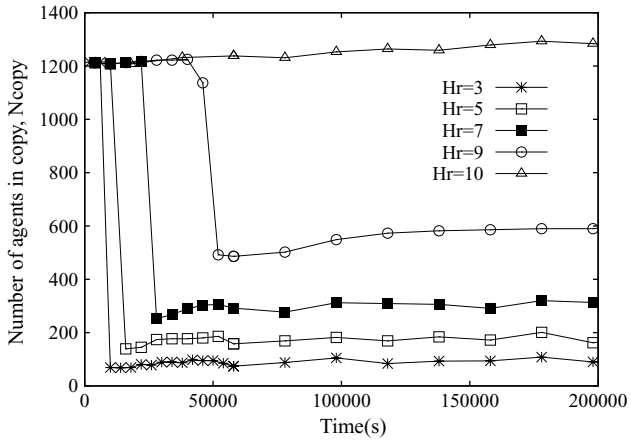

Fig. 3. Mean number of agents in *copy* mode, for different values of the pheromone threshold, H_r.

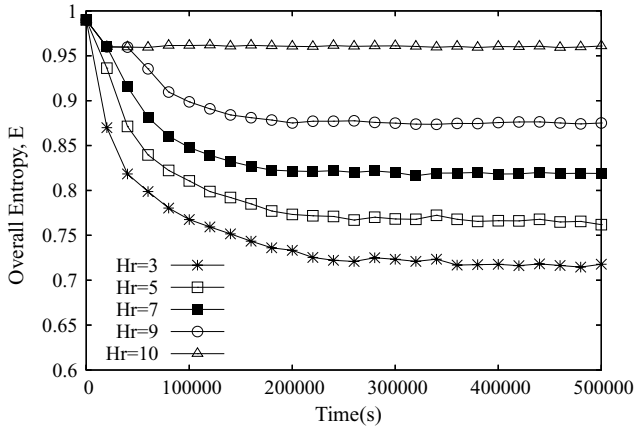

Fig. 4. Overall system entropy, for different values of the pheromone threshold, H_r.

is equal to 10 because this value can never be reached by the pheromone (see formulas (5) and (6)); hence with $H_r = 10$ all agents remain in *copy* along all their lives. After the first phase of the replication process, an equilibrium is reached because the number of new agents that are generated by Grid peers (such agents always set off in *copy* mode) and the number of agents that switch from *copy* to *move* become balanced. Moreover, if the pheromone threshold H_r is increased, the average interval of time in which an agent works in *copy* becomes longer, and therefore the average value of N_{copy} at equilibrium becomes larger.

A proper tuning of the pheromone threshold is indeed a very efficient method to enforce or reduce the generation of new replicas and the velocity and intensity of descriptor dissemination. However, a more intense dissemination is not necessarily associated with a better reorganization (to a more effective spatial separation of descriptors belonging to different classes). In fact, Figure 4 shows that lower values of the overall entropy are achieved with lower values of the

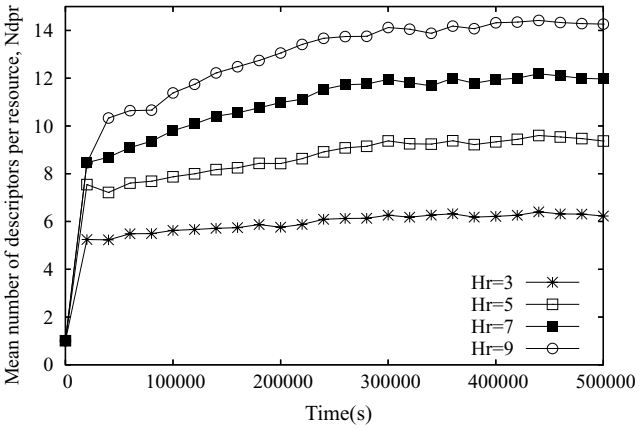

Fig. 5. Mean number of descriptors per resource, for different values of the pheromone threshold, H_r.

pheromone threshold. For example, with $H_r = 3$, the value of the overall entropy decreases from the initial value of about 1 (maximum disorder) to less than 0.72. In the opposite case, virtually no entropy decrease is observed if all the agents operate in *copy* (with $H_r = 10$), which confirms that the mode switch is strictly necessary to perform an effective spatial reorganization.

Figure 5 shows the mean number of descriptors generated per resource and confirms that descriptor dissemination is more intense if the pheromone threshold is increased, because a larger number of *copy* agents operate on the network. It can be concluded that *copy* agents are useful to replicate and disseminate descriptors but it is the *move* agents that actually perform the spatial mapping and are able to create Grid regions specialized in specific classes of resources. A balance between the two features (replication and reorganization) can be performed by appropriately tuning the pheromone threshold, H_r.

3.3 Scalability and Adaptivity of the Replication Algorithm

Another set of tests was made to evaluate several important characteristics of the replication algorithms, like its scalability and adaptivity. For these tests, the value of the threshold, H_r, was fixed at 5.0.

The algorithm is intrinsically scalable due to its self-organizing decentralized nature, since each agent operates and tunes its activeness only according to local information. To confirm this, we analyzed the processing load, L, defined as the average number of agents per second that are received, and must be processed, by a peer. Interestingly, this index only depends on the average number of agents generated by a reconnecting peer, N_{gen}, and on the frequency of their movements across the Grid, $1/T_{mov}$. Indeed, L can be obtained as follows:

$$L = \frac{N_a}{N_p \cdot T_{mov}} \approx \frac{N_{gen}}{T_{mov}}. \tag{12}$$

In the described scenario, since the average value of T_{mov} is equal to 60 seconds, and N_{gen} is set to 0.5, each peer receives and processes about one

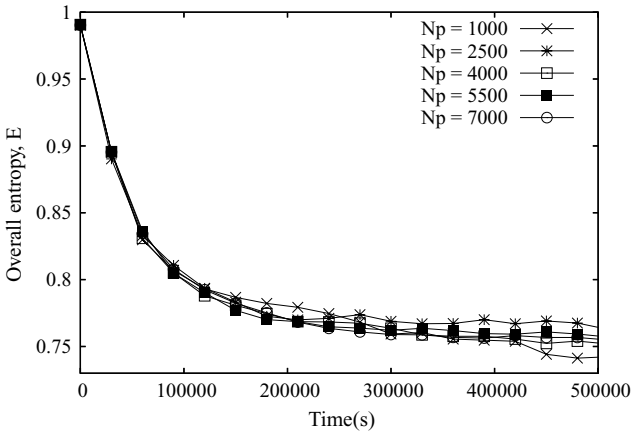

Fig. 6. Overall system entropy, for different values of the Grid size, N_p, with $H_r = 5$.

agent every 120 seconds, which can be considered an acceptable load. This theoretical result was confirmed by simulation tests. Note that the processing load does not depend on other system and algorithm parameters such as the network size, the pheromone threshold, the number of resource classes, and so on, which is a first important confirmation about the scalability of the replication algorithm.

In addition, simulations were performed with different Grid sizes. Results shown in Figure 6 show that in all the tested Grids, with N_p ranging from 1000 to 7000, the overall entropy decreases with time in a similar fashion. The values of other performance indices, including the number of descriptors per resource and the processing load, are also hardly modified by variations in the Grid size.

Another interesting type of scalability analysis pertains to the behavior of the replication algorithm versus the granularity of resource classification: So-Grid was tested for different values of the number of classes, N_c. Results pertaining to the overall entropy and the dissemination of descriptors are shown in Figures 7 and 8.

First, it can be noted that, for every tested value of N_c, the replication algorithm is always able to reduce the overall entropy and at the same time to increase the number of descriptors per resource. Moreover, analysis of steady values of these indices versus the different values of N_c, is particularly interesting. As N_c increases, lower values of entropy are achievable (Figure 7), because a finer spatial separation of resources belonging to different classes is possible. However, this is at the cost of a reduced ability to disseminate descriptors. In fact, it was observed that the drop probability assumes lower values, as it becomes increasingly difficult to find peers that hold descriptors similar to those carried by agents. Therefore, the steady value of N_{dpr} is reduced as N_c increases, as is evident in Figure 8.

A further set of tests was performed to assess the algorithms ability to adapt the mapping of descriptors to the continuous modifications of the environment. Accordingly, simulations were run with different values of the parameter

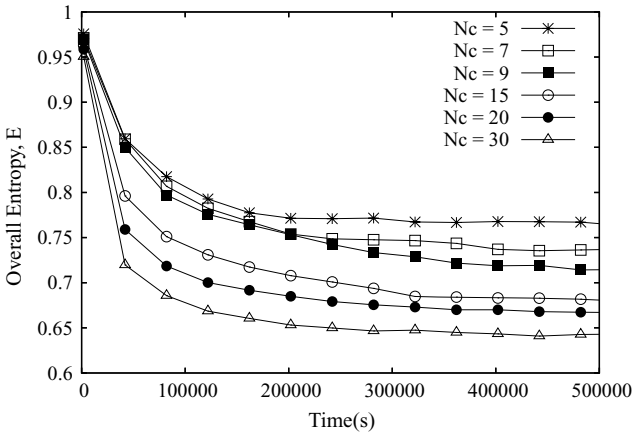

Fig. 7. Overall system entropy, for different values of the number of classes N_c, with $H_r = 5$.

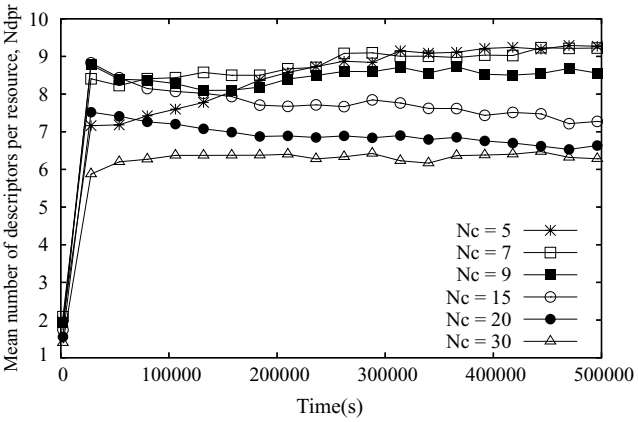

Fig. 8. Mean number of descriptors per resource, for different values of the number of classes N_c, with $H_r = 5$.

T_{peer}, the average connection time of a peer, with values ranging from 35000 to 1000000 seconds. For comparison purposes, the case in which peers never disconnect was also tested. This kind of analysis is useful because it helps clarify the mechanisms through which the information system is constructed.

Figure 9 reports the trend of the overall spatial entropy E. It appears that the work of replication agents makes this index decrease from about 1 to much lower values. After a transient phase, the value of E becomes stable: it means that the system reaches an equilibrium state despite the fact that peers go down and reconnect, agents die and others are generated, etcetera. In other words, the algorithm adapts to the varying conditions of the network and is robust with respect to them. Note that the stable value of E increases as the network becomes more dynamic (that is, with lower values of T_{peer}), because the reorganization of descriptors performed by agents is partly hindered by environmental modifications.

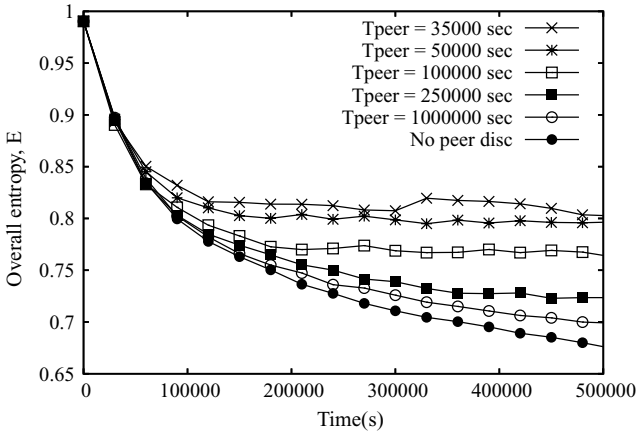

Fig. 9. Overall system entropy, for different values of the average connection time, T_{peer}.

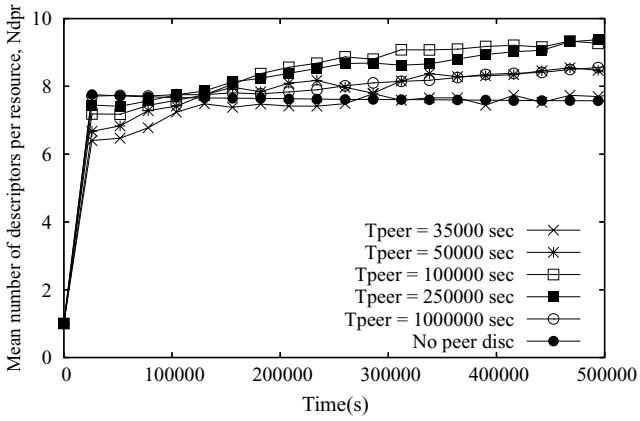

Fig. 10. Mean number of descriptors per resource, for different values of the average connection time, T_{peer}.

Figure 10 shows N_{dpr}, the average number of descriptors per resource versus the value of T_{peer}. This index increases with time in all cases, which confirms the ability of the algorithm to disseminate descriptors for different churn rates of peer. The value of N_{dpr} is actually determined by two main phenomena that work in opposite directions as the value of T_{peer} decreases. On the one hand, a lower value of T_{peer} causes the generation of a larger number of *copy* agents (of agents that operate in the copy mode), since *reconnecting* peers generate new agents, and these agents set off in the copy mode. Since copy agents are able to replicate descriptors, they tend to increase the value of N_{dpr}. On the other hand, a more frequent *disconnection* of peers tends to lower N_{dpr}, because a disconnecting peer loses all the descriptors that it has accumulated so far (see Section 2.2). The result of these two contrasting mechanisms is that the highest degree of replication is obtained for intermediate values of T_{peer}, which are more realistic on Grids. Indeed Figure 10 shows that the value of N_{dpr} first increases

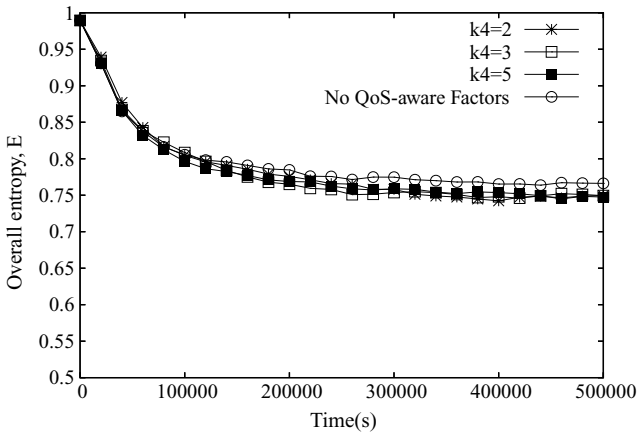

Fig. 11. Overall system entropy obtained by using the optional QoS-aware factors in functions (10) and (11), for different values of the parameter *k4*.

as T_{peer} increases from 35000 to about 100000, and then decreases for higher values of T_{peer}.

3.4 Selective Dissemination of Resource Descriptors

As discussed in Section 2.4, *enhanced* pick and drop functions (see Formulas (10) and (11)) were introduced to foster the dissemination of information concerning high QoS and/or recently updated resources, in order to enhance user satisfaction. The effectiveness of these functions was evaluated by comparing the results obtained with and without the additional factors: (i) the QoS-aware factors $FQoS_{pick}$ and $FQoS_{drop}$, defined in formulas (7) and (8), and (ii) the age-aware factor $FAge_{pick}$, defined in formula (9). Furthermore, different values of the parameters *k4* and *k5*, used in these factors, were tested.

A first interesting result is that the presence of the optional factors does not influence the overall replication process, whatever the values of *k4* and *k5*. In fact, whereas the variability of these factors depends on *k4* and *k5*, their average value is always equal to 1, which means that the average value of the enhanced pick and drop functions is not modified.

As an example, Figure 11 shows the overall entropy obtained by using the QoS-aware factors, $FQoS_{pick}$ and $FQoS_{drop}$, with different values of *k4*. It can be seen that results are not significantly modified by the use of these factors, regardless of the value of *k4*: the overall entropy is nearly the same as that obtained without using the QoS-aware factors, which is the looked-for behavior. Similar results are obtained if the age-aware factor $FAge_{pick}$ is used, regardless of the value given to the parameter *k5*.

Even though the overall process is always preserved, the *k4* parameter can be used to steer the QoS-based dissemination of descriptors. In particular, lower values of *k4* increase the variability of $FQoS_{pick}$ and $FQoS_{drop}$, and therefore allow better differentiation of the dissemination of high and low QoS descriptors.

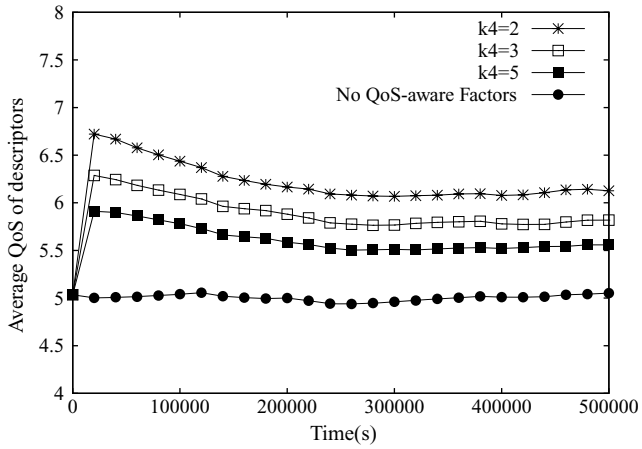

Fig. 12. Average QoS level of descriptors obtained by using the optional QoS-aware factors in functions (10) and (11), for different values of the parameter *k4*.

Figure 12 shows that, through the enhanced pick and drop functions, it is possible to significantly increase the value of $QoS_{descriptor}$, the average QoS level of the descriptors disseminated over the Grid. In fact, if optional factors are not used, which corresponds to adopting the simple pick and drop functions, the value of $QoS_{descriptor}$ remains very close to 5.0. However, by using QoS-aware factors, $QoS_{descriptor}$ actually increases with time. For example, with *k4* equal to 2, $QoS_{descriptor}$ rapidly increases in the first phase of the dissemination process, as a result of the activity of a very large number of *copy agents* (see Figure 3). Subsequently, the value of $QoS_{descriptor}$ gets stabilized at a value of about 6.1, more than 20% higher than the initial value. Notice also that the steady value of $QoS_{descriptor}$ increases as *k4* decreases, because of the higher variability of QoS-aware factors.

In a similar way, Figure 13 shows the effect obtained by using the age-aware factor $FAge_{pick}$ in formula (10). Use of this factor fosters the dissemination of descriptors of recently updated resources, which results in a notable decrease of the mean age of descriptors, $Age_{descriptor}$, depending on the value of *k5*. For example, with $k5 = 2$ the steady value of this index is about 20% lower than the value obtained without use of the age-aware factor.

In conclusion, by operating on *k4* and *k5* parameters, it is possible to tune the dissemination of high quality and/or dynamic resources on the basis of users' requirements, without significantly modifying the overall behavior of the replication algorithm. Moreover, QoS-aware and age-aware factors do not influence each other: a modification in the value of *k4* has no effect on the mean age of descriptors while a modification in *k5* does not affect the average QoS of descriptors.

4. THE SO-GRID DISCOVERY ALGORITHM

The main objectives of the replication algorithm presented and evaluated so far are the controlled replication and mapping of resource descriptors, performed

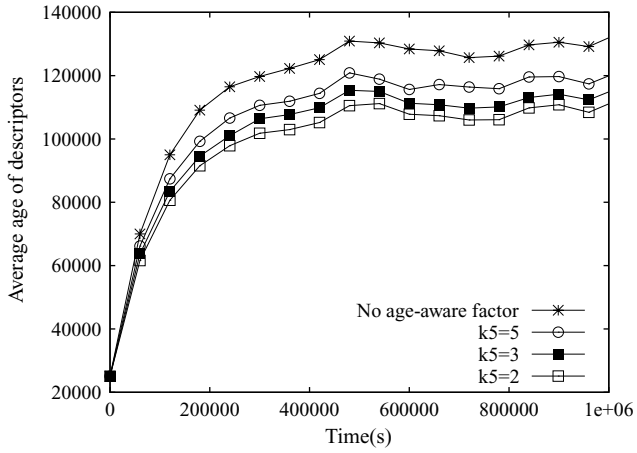

Fig. 13. Average age of descriptors obtained by using the optional age-aware factor in Function (10), for different values of the parameter *k5*.

by autonomous and self-organizing agents, and the specific dissemination of descriptors related to high-quality and recently updated resources. Whereas these objectives are valuable per se, their main benefit is to enable the use of another bio-inspired algorithm, the *discovery algorithm*, which allows users to find the most useful descriptors possible in a limited amount of time.

This is accomplished by issuing a number of query messages, which first travel the Grid in a random fashion, and then are directed towards Grid regions that have accumulated descriptors belonging to the required class; more specifically towards *representative peers* within those regions (see Figure 2). As the identification of representative peers is preliminary to the use of the discovery algorithm, the related *election algorithm* is introduced in Section 4.1; thereafter, the semi-informed discovery algorithm, which exploits a form of *marker-based stigmergy*, is described in Section 4.2.

4.1 Election of Representative Peers

As with all the other So-Grid algorithms, the election algorithm is completely decentralized, and every peer is autonomous in determining whether or not it should assume the *representative* role for a class of resources. The algorithm is performed by each peer periodically (every 2000 seconds), in three steps:

(1) The peer focuses on the class for which it stores the largest number of descriptors, say class C_i.
(2) The peer verifies whether, for class C_i, it stores an overall number of descriptors that is at least 4 times the average number of descriptors stored by a generic peer. The last value can be easily estimated, for a local region, through a simple exchange of data performed by peers. If this occurs, it means that the peer has accumulated a significant number of remote descriptors. In such a case, the peer elects itself as a *potential* representative peer for class C_i.

(3) Through a simple exchange of short-distance messages, the peer verifies if, among all the peers that are 1 and 2 hops away, there are other *potential* representative peers for the same class. In this case it compares the number of descriptors of class C_i with those peers. If the peer stores fewer descriptors than any of these neighbor peers, it abandons the election algorithm, otherwise it elects itself as a *representative peer*, and will work as an attractor for query messages issued to discover descriptors of class C_i.

The threshold used in the second step can be increased or decreased in order to, respectively, reduce or increase the number of representative peers. The goal of the third step is to avoid the election of several representative peers in the same region, which could give contrasting information to query messages.

4.2 Semi-Informed Discovery with Marker-Based Stigmergy

As mentioned in the Introduction, the So-Grid discovery algorithm is classified as *semi-informed*, as it is performed in two phases, a *blind* phase and an *informed* phase.

A discovery procedure is initiated when a user needs to find descriptors belonging to a specified class. In the *blind* phase, the requesting peer issues a number of query messages that travel the network according to the *random walk* approach [Lv et al. 2002]. The maximum number of query messages is four, but it can be limited by the number of neighbor peers that are actually online. An intermediate peer that receives a query message forwards it to a random neighbor without generating message replicas, so as to avoid flooding. In the blind phase, each query message can perform a maximum number of hops equal to a TTL (time to live) constant, which is set to seven in our study. However, the discovery operation turns to the *informed* phase as soon as a query message gets to a peer which is aware of the proximity of a representative peer and knows a route to it. In the informed phase, the message is directed towards the representative peer, where it will likely discover a large number of useful descriptors.

A query message terminates its journey either when it has performed a number of hops equal to TTL, and no representative peers have been localized, or when it arrives at a representative peer. In both cases a *queryHit* agent is generated and will return to the requesting peer following the same path in the opposite direction. In the return journey, the queryHit agent performs two kinds of operations:

(1) In the first peers of the journey, those closer to the representative peer, the queryHit agent deposits an amount of pheromone that will help successive query messages to find the same representative peer. In this article, it is assumed that a queryHit agent deposits pheromone only in the first two peers of the return path.
(2) The queryHit collects all the descriptors of the class of interest that are stored by the peers through which it passes. Such descriptors are the results of the discovery request and are delivered to the requesting peer.

The first kind of operation exploits a type of stigmergy, named *marker-based stigmergy* [Camazine et al. 2001], which is typical of different species of animals,

for example, ants and termites. The mechanism works as follows. In each peer, a pheromone base is maintained for each resource class. When a query message gets to a peer along its blind search, it checks the amount of pheromone that has been deposited in this peer for the resource class of interest. If the pheromone level exceeds a threshold H_d, it means that a representative peer is close; therefore, the discovery procedure turns to the *informed* phase and the query is driven towards the representative peer, by following the pheromone path.

As with all other So-Grid operations, the mechanism is self-organizing and can adapt to the possible change of representative peers. In particular, an evaporation mechanism is defined to assure that the pheromone deposited on a peer does not drive queries to ex-representative peers. The pheromone level at each peer is computed every two minutes. The amount of pheromone Φ_i, at the i-th time interval, is given by Formula (13). Notice that this formula is the same as that used by replication agents to control their operating mode, but uses another kind of pheromone, which is related to marker-based stigmergy instead of sematectonic stigmergy.

$$\Phi_i = Ev \cdot \Phi_{i-1} + \phi_i. \tag{13}$$

The evaporation rate, Ev, is set to 0.9; ϕ_i is equal to 1 if a pheromone deposit has been made in the last time interval by at least one agent, otherwise it is equal to 0. The pheromone level can assume values between 0 and 10. The threshold, H_d, is set to 2. With this setting, the threshold is exceeded as soon as a few queryHits deposit their pheromone at different time intervals, while the algorithm is more conservative when it has to recognize that a representative peer has been "downgraded"; up to 15 time intervals are necessary to let this level assume a value lower than the threshold.

It can happen that a peer collects pheromone deposited by queryHits coming from different representative peers of the same class. To tackle this, the peer actually maintains a different pheromone base for each neighbor and for each class. The query is forwarded to the neighbor peer associated with the higher amount of pheromone for the class of interest, provided that this maximum amount succeeds the threshold. In fact, it corresponds to sending the query to the oldest representative peer, which is most likely the representative peer that has collected the largest number of descriptors.

5. PERFORMANCE OF THE DISCOVERY ALGORITHM

Performance results related to the discovery algorithm are reported and discussed for a Grid with 2500 peers, 5 classes of resources (unless otherwise stated) and an average of 15 resources per peer. The replication and the discovery algorithms are executed concurrently: discovery requests are issued while replication agents disseminate and replicate descriptors. The parameters of the replication algorithm are set as discussed in Section 3.1. Performances are evaluated through a set of performance indices, which are reported in Table II.

The first index (average number of representative peers per class) is specifically defined to verify the effectiveness of the election algorithm presented in

Table II. Performance Indices for the So-Grid Discovery Algorithm. The Terms "stk" and "nostk" are Used when the Evaluation of the Performance Index is Restricted to Striking Queries and nonStriking Queries, Respectively

Performance Index	Value
N_{rep}	Average number of representative peers per class
P_{stk}	Percentage of striking queries
N_{res}, $N_{res}(stk)$, $N_{res}(nostk)$	Average number of discovered results per query
T_{med}, $T_{med}(stk)$, $T_{med}(nostk)$	Average response time
T_{first}, $T_{first}(stk)$, $T_{first}(nostk)$	Response time of the first result
$QoS_{results}$, $QoS_{results}(stk)$, $QoS_{results}(nostk)$	Average QoS level of results
$AGE_{results}$, $AGE_{results}(stk)$, $AGE_{results}(nostk)$	Average age of results

Section 4.1, whereas the other indices are related to the actual discovery algorithm. A *striking query* is defined as a discovery request that succeeds in reaching at least one representative peer. The number of results (the number of discovered descriptors of the class of interest) and the response times are calculated for all the requests, and separately for striking and nonstriking queries. This is done in order to evaluate the performance improvement that can be achieved through the use of representative peers and, more generically, through the dissemination of descriptors by the So-Grid replication algorithm. The response time is defined as the time elapsed between the issue of a request and the reception of the corresponding results. We calculated the average response time (the response time of a generic result) as well as the response time of the first received result. This calculation assumes that the average link delay between two adjacent peers is equal to 50 ms and the average time spent by a peer to process a query (or queryHit) message is also equal to 50 ms. Both these delays are assumed to have Gamma random distribution functions.

Another purpose is to analyze the performance of the discovery algorithm when combined with the *enhanced* pick and drop functions, which, as seen in Section 3.4, foster the dissemination of high QoS and recently updated descriptors. To this aim, we evaluated the average QoS level and the mean age of results discovered by generic discovery requests and, specifically, by striking and nonstriking queries (see Section 2.4 for the definition of age).

Figure 14 shows the average number of representative peers per class versus time along with the percentage of striking queries and the average number of results of a generic query. To depict these indices in the same chart, the average number of results is given in tens: the actual value is obtained by multiplying the depicted value by 10. Indeed, these three indices are strictly correlated: as descriptors are replicated and reorganized, more peers are elected as representative. As a consequence, more and more discovery requests (up to more than 99%) are able to reach a representative peer and, most important for the satisfaction of the user, the number of results considerably increase. Notice that all these indices increase in the first phase, and then they become almost stabilized when the system reaches a steady state. This form of "macro" stabilization does not mean that the system becomes static: on the contrary, it derives from a balancing of different dynamic phenomena, such as the disconnection and

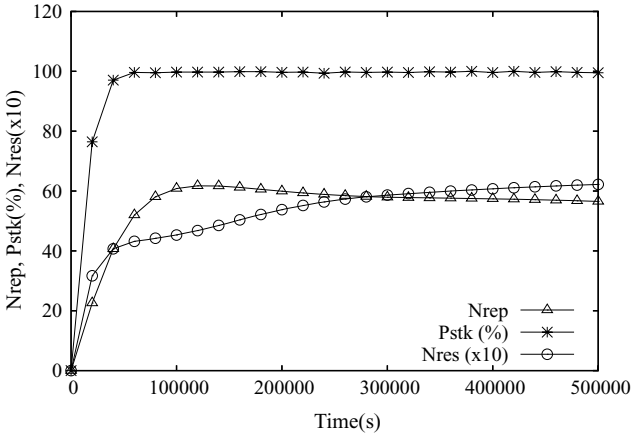

Fig. 14. Average number of representative peers per class, percentage of striking queries and mean number of results vs. time.

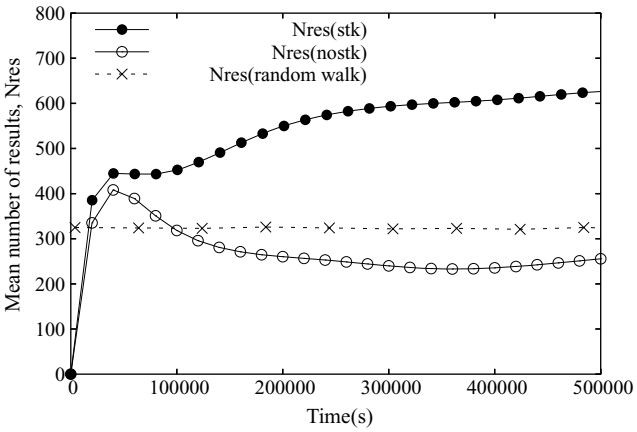

Fig. 15. Average number of results discovered by striking and nonstriking queries, and by the random walk algorithm.

reconnection of peers, the replication and deletion of descriptors, the changes in the set of representative peers and so on.

Figure 14 proves the advantage deriving from the combined use of the different So-Grid algorithms: the replication algorithm, the discovery algorithm and the auxiliary algorithms, for example, those used to elect representative peers (Section 4.1) and obtain a proper turnover of agents (Section 2.2). As further evidence, Figure 15 shows that striking queries are able to discover far more results than nonstriking queries, due to the fact that representative peers are typically located in the center of Grid regions having a higher density of useful descriptors. Since nearly all queries are actually striking in steady conditions, this results in an overall performance improvement for discovery operations. This figure also compares the results obtained with So-Grid with those obtained with the *random walk* algorithm [Lv et al. 2002], which are reported through

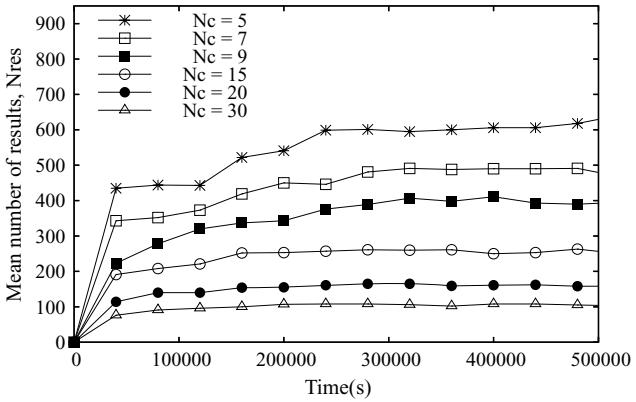

Fig. 16. Average number of results discovered by queries vs. the number of classes N_c.

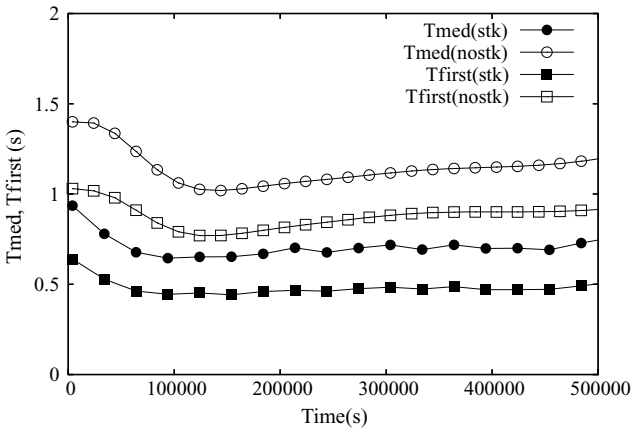

Fig. 17. Average response time and response time of the first result for striking and nonstriking queries.

the dashed line. To achieve a fair comparison, we calculated the average number of peers that are visited by a query in So-Grid, and let the random walk query travel until it visits the same number of peers. Notice that striking queries are able to collect about twice the number of results as the queries of the random walk algorithm. Even if nonstriking queries collect fewer results, this has a negligible impact on overall performance, since the percentage of nonstriking queries is very low, as seen in Figure 14.

To complete the analysis discussed in Section 3.3, Figure 16 shows the number of results versus different values of the number of classes N_c. The figure confirms that the number of results increases in all the examined cases, but it decreases as the classification of resources becomes more refined, because the fraction of resources of a single class becomes lower.

Figure 17 shows that striking queries not only discover more results but do that in much less time. Indeed, average and first result response times of striking queries are lower (if compared to respective indices for nonstriking queries)

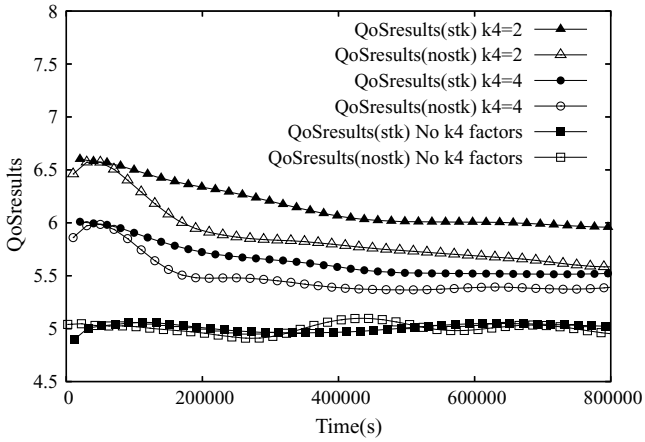

Fig. 18. Average QoS of results discovered by striking and nonstriking queries, for different values of the parameter *k4*.

and, interestingly, they further decrease as descriptors are progressively replicated and representative peers are elected. The reason is that a nonstriking query exploits the entire value of the TTL parameter, whereas a striking query terminates its path as soon as it reaches a representative peer (see Section 4.2). Clearly, this is another aspect that greatly increases user satisfaction.

A further set of tests is tailored to the evaluation of the impact that the enhanced pick and drop probability functions (introduced in Section 2.4) have on the quality and freshness of discovered results. Notice that in Section 3.4 the effect of these functions on the characteristics of generic descriptors was evaluated, whereas focus here is specifically on the descriptors returned by queryHit agents.

Figure 18 shows the average QoS of results obtained with different values of the parameter *k4*, used to tune the selective dissemination of descriptors. Performance obtained without the QoS-aware factors is labeled as "No k4 factors." It is interesting to observe that as the dissemination of "high QoS" descriptors is fostered (through the use of low values of *k4*), striking queries perform remarkably better than nonstriking queries. The reason for this is that "high QoS" descriptors are progressively accumulated on representative peers, which are discovered by striking queries. This is another confirmation that directing queries to a representative peer is indeed useful.

Finally, Figure 19 shows the average age of results, $AGE_{results}$, which are retrieved by discovery operations. By using the factor *k5*, it is possible to lower the age of results to a considerable extent. For example, results discovered with *k5* equal to 2 are around 15% fresher than results discovered without using the age-aware factor in the pick function. It is also observed that results discovered by striking queries are on average older than those discovered by nonstriking queries. The reason is that representative peers are peers that have accumulated a large number of descriptors, so it is natural that they store a number of old descriptors. Nevertheless, representative peers, by definition,

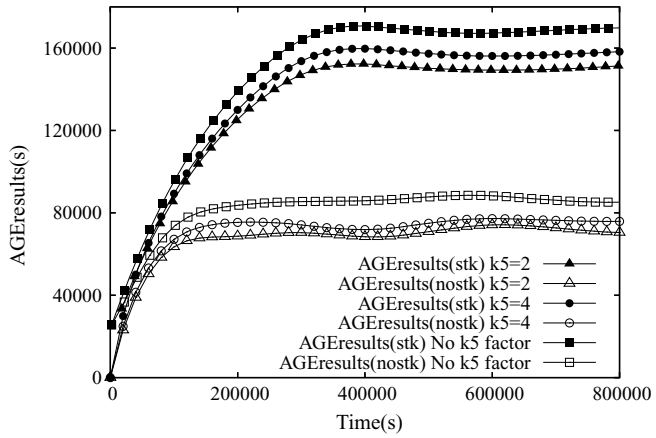

Fig. 19. Average age of results discovered by striking and nonstriking queries, for different values of the parameter *k5*.

store many more descriptors than generic peers, and allow queries to collect a larger number of results, as observed in Figure 15. Therefore, the correct way to interpret Figure 19 is that striking queries collect fresh results as well as nonstriking queries; in addition, striking queries also discover a number of less updated descriptors that can anyhow be interesting for the user. Indeed it is always up to the user the to verify that the actual resources, corresponding to the discovered descriptors, are still available and still have the characteristics advertised by the descriptors.

6. RELATED WORK

Management and discovery of resources in Grids is becoming more and more troublesome due to the dynamic characteristics of Grid hosts and resources. In most Grid frameworks deployed so far, the information system is structured according to centralized or hierarchical approaches, mostly because the client/server approach is still used today in the majority of distributed systems and in Web services frameworks. For example, Version 4 of the Globus Toolkit, the standard de facto Grid framework, is based on the Web Service Resource Framework (WSRF [The Globus Alliance 2007]). The central component in the GT4 information system is the Index Service, which collects information about Grid resources and makes this information available to users and clients. The Index Service retrieves information from multiple data sources and publishes information retrieved by other index services, giving the possibility of building the information system according to hierarchical, decentralized or hybrid architectures, even though the hierarchical model is still the most frequently used [Schopf et al. 2005].

Nowadays, the research and development community agrees that the centralized approaches are becoming unbearable, since they create administrative bottlenecks and hence are not scalable. Conversely, the adoption of decentralized approaches, like the P2P paradigm, can favor Grid scalability [Iamnitchi and Foster 2003; Taylor 2004]. A hierarchical information system can be viable

within a single Organization or in a small-scale Grid, but it can become impractical in a large multi-institutional Grid for several reasons, among which:

—fault-tolerance is limited by the presence of a bottleneck at the tree root;
—a significant amount of memory space must be reserved in high level information servers to store information about a large number of resources, limiting the scalability of the Grid;
—information servers belonging to different levels must carry very different computation and traffic loads, which leads to challenging problems concerning load imbalance;
—the hierarchical organization can hinder the autonomous administration of different Organizations.

Novel approaches for Grid management, and in particular for the construction of a scalable and efficient information system, should have the following properties: (i) self-organization (meaning that Grid components are autonomous and do not rely on any external supervisor), (ii) decentralization (decisions are to be taken only on the basis of local information), and (iii) adaptivity (mechanisms must be provided to cope with dynamic characteristics of hosts and resources).

Requirements and properties of Self Organizing Grids are sketched in Erdil et al. [2005]. Some of the issues presented in that paper are concretely applied in this work: for example, clustering of resources in order to facilitate discovery operations, election of representative nodes within clusters, and adaptive dissemination of information. A self-organization mechanism is proposed in Padmanabhan et al. [2005] to classify Grid nodes in a number of groups based on the choice of specific similarity characteristics. Each group elects a leader node that receives requests tailored to the discovery of resources that are likely to be maintained by such a group. This is an interesting approach but it still has nonscalable characteristics: for example, it is required that each Grid node has a link to all leader nodes, which is clearly problematic in a very large Grid. A self-organizing mechanism is exploited in Chakravarti et al. [2005] to build an adaptive overlay structure for the execution of a large number of tasks in a Grid.

Similarly to the latter work, the So-Grid algorithms proposed in this article exhibit several characteristics of both biological systems and mobile agent systems (MAS), as discussed in the Introduction. In many aspects So-Grid is specifically inspired by ant algorithms, a class of agent systems which aim to solve very complex problems by imitating the behavior of some species of ants [Bonabeau et al. 1999].

The Anthill system [Babaoglu et al. 2002] is tailored to the design, implementation and evaluation of P2P applications based on multiagent and evolutionary programming. It is composed of a collection of interconnected *nests*. Each nest is a peer entity that makes its storage and computational resources available to swarms of *ants*, mobile agents that travel the Grid to satisfy user requests. However, while in Anthill, ants are generated after user requests, in So-Grid agents operate continuously and autonomously, since the reorganization of

descriptors must be performed prior to user requests. One Anthill application is Messor [Montresor et al. 2002], which is aimed at supporting the concurrent execution of a large set of independent jobs. In Messor the objective of ants is to find *overloaded* and *underloaded* nests, and then assign jobs in order to improve load balance among nests. In a similar way, in Cao [2004], agents that are representative of different local schedulers use ant algorithms to balance the load of these schedulers, so achieving an improved job scheduling pattern at a global level.

The stigmergy paradigm, through the definition of techniques based on ant pheromone, is exploited to drive the behavior of So-Grid agents, making them able to take actions autonomously, without having an overall view of the system. This kind of approach is discussed in Van Dyke Parunak et al. [2005], where a decentralized scheme, inspired by insect pheromone, is used to limit the activity of a single agent when it is no longer acting to accomplish the system goal.

The two main objectives of So-Grid are the replication and dissemination of metadata documents and the discovery of resources. These issues are obviously correlated, since an intelligent dissemination can facilitate discovery. Nevertheless, information dissemination is also functional for other important requirements, such as fault tolerance, load balancing, reduced access latency and bandwidth consumption. Information dissemination is indeed a fundamental and frequently occurring problem in large, dynamic and distributed systems. In Cohen and Shenker [2002], the authors examine a number of techniques that can improve the effectiveness of blind search by proactively replicating data. In particular, two natural but very different replication strategies are described: uniform and proportional. The uniform strategy, replicating everything equally, appears naive, whereas the proportional strategy, where more popular items are more replicated, is designed to perform better but fails to do so. Actually, it is shown that any strategy that lies between the two, performs better than the two extreme strategies. Iamnitchi and Foster [2005], proposed disseminating information selectively to groups of users with common interests, so that data is sent only to where it is wanted. In our article, instead of classifying users, the proposal is to exploit the classification of resources: resource metadata documents are replicated and disseminated with the purpose of creating regions of the network that are specialized in specific classes of resources. In Aktas et al. [2007] information dissemination is combined with the issue of effective replica placement, since the main interest is to place replicas in the proximity of requesting clients by taking into account changing demand patterns. A metadata document is replicated if its demand is higher than a defined threshold and each replica is placed according to a multicast mechanism that aims to discover the data system which is the closest to demanding clients.

An additional objective of the present work is the selective dissemination of information pertaining to high QoS and dynamic resources. Interesting papers that cope with these issues are Ran [2003], Vu et al. [2005], and Cheema et al. [2005], which are also referenced in the Introduction and in Section 2.4.

As for the resource discovery issue, the So-Grid algorithm is in the research category of P2P resource discovery procedures, discussed in Trunfio et al. [2007]. P2P algorithms can be categorized as *structured* or *unstructured*. Structured

protocols, based on highly structured overlays and distributed hash tables (e.g. Chord [Stoica et al. 2001]), are usually efficient in file sharing P2P networks, but structure management can be cumbersome and poorly scalable in large and dynamic Grids. Therefore, unstructured protocols seem to be preferable in Grids. Unstructured algorithms can be further classified into *blind* and *informed* [Tsoumakos and Roussopoulos 2003b]. If nodes have no information about where the resources are actually located, a search request is performed through a random exploration of the network; therefore a *blind* search mechanism is adopted, such as "flooding" or "random walk" [Lv et al. 2002]. If a centralized or distributed information service maintains information about resource location, it is possible to drive search requests through an *informed* mechanism, such as "routing indices" [Crespo and Garcia-Molina 2002] or "adaptive probabilistic search" [Tsoumakos and Roussopoulos 2003a].

The So-Grid discovery algorithm can be categorized as *semi-informed*, since it combines the benefits of both *blind* and *informed* resource discovery approaches. In fact, a pure blind approach is simple to implement but has limited performance and can cause an excessive network load, whereas a pure informed approach requires a structured resource organization, which is generally impractical in a large, heterogeneous, and dynamic Grid.

So-Grid assumes the pre-existence of an algorithm for the classification of resources. This assumption is common in similar works: in Crespo and Garcia-Molina [2002], performance of a discovery technique is evaluated by assuming that resources have been previously classified in four disjoint classes. Classification can be made by characterizing the resources with a set of parameters that can have discrete or continuous values. Classes can be determined with the use of Hilbert curves that represent the different parameters on one dimension [Andrzejak and Xu 2002]; alternatively, an n-dimension distance metric can be used to determine the similarity among resources [Kronfol 2002].

7. CONCLUSIONS

This article presents So-Grid, a set of algorithms for the construction of a decentralized Grid information system that features profitable characteristics such as: self-organization, since system entities are autonomous, and there is no central control; adaptive behavior, which allows a prompt reaction to the modifications of a dynamic environment; scalability, since the So-Grid algorithms are not biased by the Grid size. These characteristics derive from the operations of very simple agents, whose behavior is inspired by biological systems, in particular ant and termite colonies. Two different kinds of agents operate continuously and concurrently: some replicate and reorganize resource descriptors on the Grid, while the others perform resource discovery operations on behalf of users.

Simulation showed that the So-Grid replication algorithm is effective in the controlled propagation and reorganization of information. This permits the definition of a semi-informed discovery algorithm that is able to find useful resources in a short time. The discovery algorithm drives message queries to Grid representative peers, which are elected within Grid regions that store a large

number of resource descriptors having specific characteristics. An enhanced version of the replication algorithm is defined to foster the dissemination, and facilitate the discovery, of recently updated information and of high quality resources, in the case where there is a common agreement about the evaluation of resource quality.

The So-Grid algorithms can be used in any Grid network in which (i) connections among hosts are deployed with a P2P overlay and (ii) users, to build their distributed applications, need to discover hardware and software resources belonging to given categories, so that they can choose those that are the most appropriate for their requirements. Moreover, the Web Services approach, which is adopted in modern Grid frameworks, like the Globus Toolkit, is particularly suitable to support the multiagent approach of So-Grid. Indeed, agent movements across the Grid can be handled as invocations among Web Services that are exposed by Grid hosts.

REFERENCES

AKTAS, M. S., FOX, G. C., AND PIERCE, M. 2007. Fault tolerant high performance information services for dynamic collections of grid and Web services. *Fut. Gen. Comput. Syst. 23,* 3, 317–337.

ANDRZEJAK, A. AND XU, Z. 2002. Scalable, efficient range queries for grid information services. In *Proceedings of the Second IEEE International Conference on Peer-to-Peer Computing (P2P'02)*. IEEE Computer Society, Washington, DC, 33–40.

BABAOGLU, O., MELING, H., AND MONTRESOR, A. 2002. Anthill: A framework for the development of agent-based peer-to-peer systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*. IEEE Computer Society, Washington, DC, 15–22.

BARABÁSI, A.-L. AND ALBERT, R. 1999. Emergence of scaling in random networks. *Science 286,* 5439 (Oct.), 509–512.

BONABEAU, E., DORIGO, M., AND THERAULAZ, G. 1999. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, NY.

CAMAZINE, S., FRANKS, N. R., SNEYD, J., BONABEAU, E., DENEUBOURG, J.-L., AND THERAULA, G. 2001. *Self-Organization in Biological Systems*. Princeton University Press, Princeton, NJ.

CAO, J. 2004. Self-organizing agents for grid load balancing. In *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing GRID'04*. IEEE Computer Society, Pittsburgh, 388–395.

CHAKRAVARTI, A. J., BAUMGARTNER, G., AND LAURIA, M. 2005. The organic grid: self-organizing computation on a peer-to-peer network. *IEEE Trans. Syst., Man, and Cyber. Part A 35,* 3, 373–384.

CHEEMA, A. S., MUHAMMAD, M., AND GUPTA, I. 2005. Peer-to-peer discovery of computational resources for grid applications. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*. Seattle, WA, 179–185.

COHEN, E. AND SHENKER, S. 2002. Replication strategies in unstructured peer-to-peer networks. In *Proceedings of the Special Interest Group on Data Communication (ACM SIGCOMM'02)*. Pittsburgh, Pennsylvania.

CRESPO, A. AND GARCIA-MOLINA, H. 2002. Routing indices for peer-to-peer systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*. 23–33.

DASGUPTA, P. 2004. Intelligent agent enabled peer-to-peer search using ant-based heuristics. In *Proceedings of the International Conference on Artificial Intelligence (IC-AI'04)*. 351–357.

DENEUBOURG, J. L., GOSS, S., FRANKS, N., SENDOVA-FRANKS, A., DETRAIN, C., AND CHRÉTIEN, L. 1990. The dynamics of collective sorting robot-like ants and ant-like robots. In *Proceedings of From Animals to Animats: The First International Conference on the Simulation of Adaptive Behavior*. MIT Press, Cambridge, MA, USA, 356–363.

ERDIL, D. C., LEWIS, M. J., AND ABU-GHAZALEH, N. 2005. An adaptive approach to information dissemination in self-organizing grids. In *Proceedings of the International Conference on Autonomic and Autonomous Systems (ICAS'06)*. Silicon Valley, CA.

FORESTIERO, A., MASTROIANNI, C., AND SPEZZANO, G. 2005. Construction of a peer-to-peer information system in grids. In *Self-Organization and Autonomic Informatics (I)*, H. Czap, R. Unland, C. Branki, and H. Tianfield, Eds. Frontiers in Artificial Intelligence and Applications, vol. 135. IOS Press, Amsterdam, The Netherlands, 220–236.

FORESTIERO, A., MASTROIANNI, C., AND SPEZZANO, G. 2006. An agent based semi-informed protocol for resource discovery in grids. In *Proceedings of the International Conference on Computational Science(4) (ICCS'06)*. 1047–1054.

FOSTER, I. AND KESSELMAN, C. 2003. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA.

GRASSE', P. 1959. La reconstruction du nid et les coordinations inter-individuelles chez belicositermes natalensis et cubitermes sp. la thorie de la stigmergie: Essai d'interpretation du comportement des termites constructeurs. *Insectes Sociaux 6*, 41–84.

IAMNITCHI, A. AND FOSTER, I. 2003. On death, taxes, and the convergence of peer-to-peer and grid computing. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*. Berkeley, CA.

IAMNITCHI, A. AND FOSTER, I. 2005. Interest-aware information dissemination in small-world communities. In *Proceedings of the 14th IEEE International Symposium on High Performance Distributed Computing, (HPDC)*. Research Triangle Park, NC, USA.

IAMNITCHI, A., FOSTER, I., WEGLARZ, J., NABRZYSKI, J., SCHOPF, J., AND STROINSKI, M. 2003. A peer-to-peer approach to resource location in grid environments. In *Grid Resource Management*. Kluwer Publishing.

KRONFOL, A. Z. 2002. *FASD: A Fault-tolerant, Adaptive, Scalable, Distributed Search Engine*. PhD dissertation, at http://citeseer.ist.psu.edu/571354.html.

LV, Q., CAO, P., COHEN, E., LI, K., AND SHENKER, S. 2002. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th International Conference on Supercomputing (ICS'02)*. ACM Press, New York, NY, 84–95.

MARTIN, M., CHOPARD, B., AND ALBUQUERQUE, P. 2002. Formation of an ant cemetery: Swarm intelligence or statistical accident? *Fut. Gen. Comput. Syst. 18,* 7, 951–959.

MONTRESOR, A., MELING, H., AND MONTRESOR, A. 2002. Messor: Load-balancing through a swarm of autonomous agents. In *International Workshop on Agents and Peer-to-Peer Computing*. Bologna, Italy.

PADMANABHAN, A., WANG, S., GHOSH, S., AND BRIGGS, R. 2005. A self-organized grouping (sog) method for efficient grid resource discovery. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*. Seattle, WA.

PETERSEN, K., SPREITZER, M. J., TERRY, D. B., THEIMER, M. M., AND DEMERS, A. J. 1997. Flexible update propagation for weakly consistent replication. In *Proceedings of the Sixteenth ACM Symposium on Operating Systems Principles (SOSP'97)*. ACM Press, New York, NY, 288–301.

RAN, S. 2003. A model for Web services discovery with qos. *ACM SIGecom Exch. 4,* 1, 1–10.

SCHOPF, J. M., D'ARCY, M., MILLER, N., PEARLMAN, L., FOSTER, I., AND KESSELMAN, C. 2005. Monitoring and discovery in a Web services framework: Functionality and performance of the globus toolkit's mds4. Tech. Rep. ANL/MCS-P1248-0405, Argonne National Laboratory. April.

SHARMA, P., ESTRIN, D., FLOYD, S., AND JACOBSON, V. 1997. Scalable timers for soft state protocols. In *Proceedings of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies, (INFOCOM'97)*. Vol. 1. IEEE Computer Society, Washington, DC, 222–229.

STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. 2001. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'01)*. ACM Press, New York, NY, 149–160.

SYCARA, K. 1998. Multiagent systems. *Artif. Intell. Mag. 10,* 2, 79–93.

TAYLOR, I. J. 2004. *From P2P to Web Services and Grids: Peers in a Client/Server World*. Springer.

THE GLOBUS ALLIANCE. 2007. The Web services resource framework, http://www.globus.org/wsrf/.

TRUNFIO, P., TALIA, D., PAPADAKIS, H., FRAGOPOULOU, P., MORDACCHINI, M., PENNANEN, M., POPOV, K., VLASSOV, V., AND HARIDI, S. 2007. Peer-to-peer resource discovery in grids: Models and systems. *Fut. Gen. Comput. Syst. 23,* 7 (Aug.), 864–878.

TSOUMAKOS, D. AND ROUSSOPOULOS, N. 2003a. Adaptive probabilistic search for peer-to-peer networks. In *Proceedings of the Third IEEE International Conference on P2P Computing (P2P'03)*. 102–109.

TSOUMAKOS, D. AND ROUSSOPOULOS, N. 2003b. A comparison of peer-to-peer search methods. In *Proceedings of the 6th International Workshop on the Web and Databases (WebDB'03)*. San Diego, CA, 61–66.

VAN DYKE PARUNAK, H., BRUECKNER, S., MATTHEWS, R. S., AND SAUTER, J. A. 2005. Pheromone learning for self-organizing agents. *IEEE Trans. Syst. Man, Cyber. Part A 35,* 3, 316–326.

VU, L.-H., HAUSWIRTH, M., AND ABERER, K. 2005. QoS-based service selection and ranking with trust and reputation management. In *Proceedings of the International Conference on Cooperative Information Systems (CoopIS'05), 31 Oct.–4 Nov. 2005, Agia Napa, Cyprus.* Vol. 3760. 446–483.

Received April 2007; revised February 2008; accepted February 2008