A Swarm Algorithm for a Self-Structured P2P Information System

Agostino Forestiero, *Member, IEEE,* and Carlo Mastroianni, *Member, IEEE*

***Abstract*— This paper introduces Antares, which is a bio-inspired algorithm for the construction of a decentralized and self-organized P2P information system in computational grids. This algorithm exploits the properties of *ant* 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. Here, the work of ant-inspired agents is tailored to the controlled replication and relocation of "descriptors," that is, documents that contain metadata information about grid resources. Agents travel the grid through P2P interconnections, and replicate and spatially sort descriptors so as to accumulate those represented by identical or similar indexes into neighbor grid hosts. The resulting information system is here referred to as *self-structured*, because it exploits the self-organizing characteristics of ant-inspired agents, and the association of descriptors with hosts is not predetermined but adapts to the varying conditions of the grid. This self-structured organization combines the benefits of both *unstructured* and *structured* P2P information systems. Indeed, being basically unstructured, Antares is easy to maintain in a dynamic grid, in which joins and departs of hosts can be frequent events. On the other hand, the aggregation and spatial ordering of descriptors can improve the rapidity and effectiveness of discovery operations, which is a beneficial feature typical of structured systems. Performance analysis proves that ant operations allow the information system to be efficiently reorganized, thus improving the efficacy of both simple and range queries.**

***Index Terms*— Ant algorithms, grid, information dissemination, information system, peer-to-peer (P2P), resource discovery.**

I. Introduction

GRID COMPUTING [1] 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 [2]), resources are offered through the invocation of web services, which boast of enriched functionalities such as lifecycle and state management.

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

Owing to the inherent scalability and robustness of peer-to-peer (P2P) algorithms, several P2P approaches have recently been proposed for resource organization and discovery in grid environments [3], [4]. The ultimate goal of these approaches is to allow users to rapidly locate grid resources or services, either hardware or software, which have the required characteristics; this is generally reduced to the problem of finding related *descriptors*, through which it is possible to access the corresponding resources. A descriptor may contain a syntactical description of the resource/service [i.e., a WSDL (web services description language) document] and/or an ontology description of resource/service capabilities. In a grid, after issuing a query, users can discover a number of descriptors of possibly useful resources, and then can choose the resources that are the most appropriate for their purposes.

In P2P systems, descriptors are often indexed through bit vectors, or *keys*, which can have two different meanings. The first is that each bit represents the presence or absence of a given *topic* [5], [6]: this method is particularly appropriate if the resource of interest is a document, because it is possible to define the different topics on which this document focuses. Alternatively, a resource or service can be mapped by a hash function into a binary key. If the hash function is locality preserving, as for example in [7], [8], similar descriptor keys are assigned to resources having similar characteristics.

In this paper, Ant-Based Algorithm for Resource (Antares) management in grids, which was formerly presented in [9], is proposed as an approach for the construction of a grid information system, which is inspired by the behavior of some species of ants [10]. The Antares algorithm is able to disseminate and reorganize descriptors and, as a consequence of this, it facilitates and speeds up discovery operations. More specifically, Antares agents concurrently achieve multiple objectives: 1) they *replicate* and *disseminate* descriptors, thus spreading useful information on the grid; 2) they *spatially sort* descriptors, so that descriptors indexed by similar keys are placed in neighbor hosts, which greatly facilitates discovery operations; 3) thanks to the self-organizing nature of the ant-based approach, the agents adapt the reorganization of descriptors to the ever-changing environment, for example to the joinings and departures of grid hosts and to the changing characteristics of resources.

Manuscript received January 4, 2008; revised June 9, 2008; accepted November 13, 2008. Current version published August 14, 2009. This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

The authors are with the CNR Institute for High Performance Computing and Networks, 87036 Rende (CS), Italy (e-mail: forestiero@icar.cnr.it; mastroianni@icar.cnr.it).

Digital Object Identifier 10.1109/TEVC.2008.2011478

The Antares approach can be positioned along a well-known research avenue whose objective is to devise possible applications of nature inspired algorithms [11], in particular of *ant algorithms*, i.e., algorithms inspired by the behavior of ants [10], [12]. We recently proposed an approach for the reorganization and discovery of resources that are preclassified in a given number of classes [13], [14]. This enables the creation of grid regions specialized in a specific resource type, which improves the performance of discovery operations but does not allow for the spatial sorting of descriptors, thus limiting the efficiency of simple and, in addition, range queries. Conversely, Antares is specifically designed to tackle the case in which the indexes of resource descriptors are bit vectors. Since similar resources are assigned similar descriptor indexes, it is possible to define a similarity measure among resources, through the comparison of related indexes.

Antares is partly inspired by the work of Lumer and Faieta [15], who devised a method to spatially sort data items through the operations of simple robots. As in [15], the reorganization of descriptors in Antares is achieved by means of ant *pick* and *drop* operations, which are driven by corresponding *pick* and *drop probability functions*. However, the application domain of Antares is completely different, so the basic approach of Lumer and Faieta has been adapted to our purposes in several ways: 1) descriptors are not only sorted, as in [15], but also *replicated*, in order to disseminate useful information on the grid and facilitate search requests. These two objectives are achieved by the definition of two different *modes* of agents, which are called *copy* and *move*; 2) Each cell/host can contain a number of descriptors, not just one item as in [15], thus enabling the accumulation of descriptors on grid hosts; 3) Since Antares operates in a distributed computing environment, agents limit the frequency and range of their P2P hops in order to reduce the network and computing load; 4) A discovery protocol is defined to exploit the obtained rearrangement and spatial ordering of descriptors.

P2P information systems are usually classified into *unstructured* and *structured* [16]. The grid information system constructed with Antares is basically *unstructured*, in the sense that descriptors are not required to be mapped onto specified hosts, but they are freely placed by ants through their pick and drop operations. This ensures valuable features such as self-organization, adaptivity, and, as a consequence, scalability. Conversely, in *structured* approaches, resources are assigned to hosts with a well-specified strategy, for example through a distributed hash table (DHT) [17], by which the correct location of a resource on the network is obtained as the result of a hash function. The structured approach generally speeds up discovery operations but also presents important drawbacks [18], such as: 1) limited fault tolerance and scalability, since the disconnection of a peer requires an immediate reorganization of resources and of peer index tables; and 2) poor load balancing, because peers that are assigned the descriptors of the most popular resources can be easily overloaded.

Though being basically unstructured, Antares features a self-emerging organization of descriptors, through which some *structured* properties emerge, since descriptors are aggregated and spatially sorted on the basis of their indexes. Therefore, the resulting information system is referred to as *self-structured*, because it exploits the self-organizing characteristics of ant-inspired agents, and the association of descriptors with hosts is not predetermined but adapts to the varying conditions of the grid. Owing to these characteristics, Antares retains important benefits that are typical of structured systems. In particular, it enables the use of an *informed* discovery algorithm, through which a query message, which is issued to discover resources indexed by a specified *target* descriptor, can efficiently explore the grid and collect information about such resources. The discovery algorithm is based on a best neighbor approach: at each step, it drives the query message towards the neighbor peer that possesses descriptors which are the most similar to the target descriptor. Since descriptors have been spatially sorted by Antares agents, in many cases this algorithm allows the query to reach a grid region in which several useful descriptors have been accumulated.

Moreover, as opposed to unstructured P2P systems, and also to many structured ones, Antares naturally supports *range queries*, that is, queries issued to discover resources defined with relaxed constraints. In Antares, a range query is managed by defining a target descriptor in which one or more bits are "wildcard bits" that can assume either the value 0 or 1. In this case, the best neighbor is chosen by masking these bits and considering only the bits that are given a specified value. Thanks to the spatial sorting of descriptors, also a range query can be directed to descriptors that match the target descriptor, in both valued and wildcard bits.

The rest of the paper is organized as follows. Section II discusses related work, and Section III describes the Antares algorithm. Section IV shows that the Antares algorithm succeeds in the spatial replication and sorting of descriptors. In fact, event-based simulation proves that agents successfully generate and disseminate descriptor replicas, and at the same time that the homogeneity of co-located or nearby located descriptors is notably increased, meaning that descriptors are effectively reorganized and sorted on the grid. In Section V, the performance analysis also proves the Antares's ability to resolve efficiently both simple and range queries. Section VI concludes the paper.

II. Related Work

In most grid systems, resource discovery is implemented 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 service-based frameworks. However, a hierarchical resource discovery service is viable within a single organization or in a small-scale grid, but it becomes impractical in a large multi-institutional grid for several reasons [19], among which are the following.

1) Scalability is limited because a significant amount of memory space must be reserved at the tree root node to maintain information about a large number of resources.
2) Information servers belonging to different levels of the hierarchy must carry very different computation and traffic loads, which leads to challenging problems concerning load imbalance.

3) The hierarchical organization can hinder the autonomous administration of servers.
4) Fault-tolerance is limited by the presence of a single point of failure at the root of the hierarchy.

In the last few years, the P2P paradigm has emerged as an alternative to centralized and hierarchical approaches. Novel approaches for the construction of scalable and efficient grid information systems need to have the following properties [3], [4], [20]: self-organization (meaning that grid components are autonomous and do not rely on any external supervisor), decentralization (decisions are to be taken only on the basis of local information), and adaptivity (mechanisms must be provided to cope with the dynamic characteristics of hosts and resources).

Requirements and properties of "self-organizing grids" are sketched in [21]. Some of the issues presented in this paper are concretely applied in this paper: for example, reorganization of resources in order to facilitate discovery operations and adaptive dissemination of information. Another self-organization mechanism is proposed in [22] to classify grid nodes in groups on the basis of a similarity measure. Each group elects a leader node that receives requests tailored to the discovery of resources which are likely to be maintained by such 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 the leader nodes, which is clearly problematic in a very large grid. A self-organizing mechanism is also exploited in [23] to build an adaptive overlay structure for the execution of a large number of tasks in a grid.

Similar to the latter work, the Antares algorithm presented in this paper exhibits several characteristics of mobile agent systems (MAS) and biological systems. Indeed, many biological systems can be quite naturally emulated in a distributed system through the multiagent paradigm ([24]): for example, 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, self-organization, decentralization, and adaptivity. Coordination among agents is essential to improve the effectiveness of their tasks, in particular for resource discovery. In agent-based systems, communication is performed either with direct communication among agents, as in the *UWAgents* system [25], or with agents migration as in the *MAG* (mobile agents for grid computing environments) middleware [26]. Conversely, Antares exploits the *stigmergy* paradigm [27], since agents interact and cooperate through the modifications of the environment that are induced by their operations. In fact, the behavior of an agent is driven by the state of the local region of the grid, which in turn is modified by the operations of other agents.

Antares is specifically inspired by ant algorithms, a class of agent systems that can solve very complex problems by imitating the behavior of some species of ants [10]. Among such systems, Anthill [28] 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 Antares, agents operate continuously and autonomously, since the reorganization of descriptors must be performed prior to user requests. Antares agents are simple and perform simple operations, but a sort of "swarm intelligence" emerges from their collective behavior.

The two main objectives of Antares 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 to 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 [29], 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, which replicates everything equally, appears naïve, whereas the proportional strategy, where more popular items are more frequently 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. In [30] it is proposed to disseminate information selectively to groups of users with common interests, so that data is sent only to where it is wanted. In this paper, 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 [31] 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. Specifically, 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 server which is the closest to demanding clients. Antares differs from these and similar algorithms in that its aim is not only the dissemination but also the reorganization of descriptors, in order to expedite resource discovery procedures. Accordingly, in Antares the replication algorithm is driven by probabilistic choices that cluster and spatially sort descriptors over the network.

As for the *resource discovery* issue, Antares puts itself along the research avenue of P2P resource discovery algorithms. Structured P2P algorithms are usually efficient in file-sharing P2P networks, but structure management can be cumbersome and poorly scalable in large and dynamic grids, especially when the churn rate, i.e., the frequency of peer disconnections, is high. Therefore, unstructured protocols seem to be preferable in grids. Unstructured algorithms can be further classified into *blind* and *informed* [32]. If nodes have no information on 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 the "random walk" [33]. 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 the "routing indices" [5] or the "adaptive probabilistic search" [34]. The Antares discovery algorithm is *informed*, since queries are directed towards peers that contain descriptors more and more similar to the target descriptors. However, this is not achieved with a centralized approach, but messages are forwarded only on the basis of local information. It is the spatial sorting of descriptors that guarantees a good chance of success, i.e., a high probability of driving queries to grid hosts that maintain a consistent number of target descriptors.

Another main objective of Antares is to serve *range queries* efficiently. The efficient management of these queries is a fundamental requirement in grid systems, as users often do not need to discover well-defined resources but resources having more loosely specified characteristics. For example, a query might be issued to discover a supercomputer with CPU speed comprised in a given range, or a web service whose estimated service time is within a given interval.

A naïve way to manage range queries is to issue as many simple queries as are sufficient to cover all the possible values of target descriptors, and then collect the results. This solution is clearly inefficient, since it does not exploit the similarity of the target descriptors specified in the different simple queries. The efficient execution of range queries is indeed a very tough issue for grids and P2P systems [35], [36]. This problem is almost impractical for unstructured systems, since similar resources are stored on different peers without any pattern or structure that could facilitate the search procedure. Some types of structured systems are capable of serving range queries, but often at the cost of either maintaining complex tree-like structures [8] or increasing the traffic load by issuing a number of sub-queries [37]. The Squid discovery protocol [38] uses a dimension-reducing technique, i.e., the space filling curve (SFC), to map multi-attribute keywords to a one-dimensional space, i.e., a ring structure similar to that used by Chord. This enables Squid to support queries defined through partial keywords, wildcards, and ranges. However, SFCs have an important drawback in that close points in the 1D structure are mapped from close points in the multi-attribute space, but the reverse is not true. A sub-cube in the multi-attribute space can be mapped to very distant points that belong to different ring segments. In the case of a range query, the Squid system must then forward query messages to all the segments of the ring that cover the sub-cube of the original multi-attribute space which is defined by the range query.

Though being basically unstructured, the Antares self-structured information system supports range queries. In fact, since descriptors indexed by similar keys are likely to be located in neighbor hosts, a single range query can discover and collect several target descriptors in the same region of the grid. This feature will be discussed at the end of the performance evaluation section. To the best of our knowledge, Antares is the first effort to serve range queries in a nonstructured P2P system, thus avoiding all the problems and limitations, which are already mentioned in the Introduction and in this section, which are inherited by the structured P2P approach.

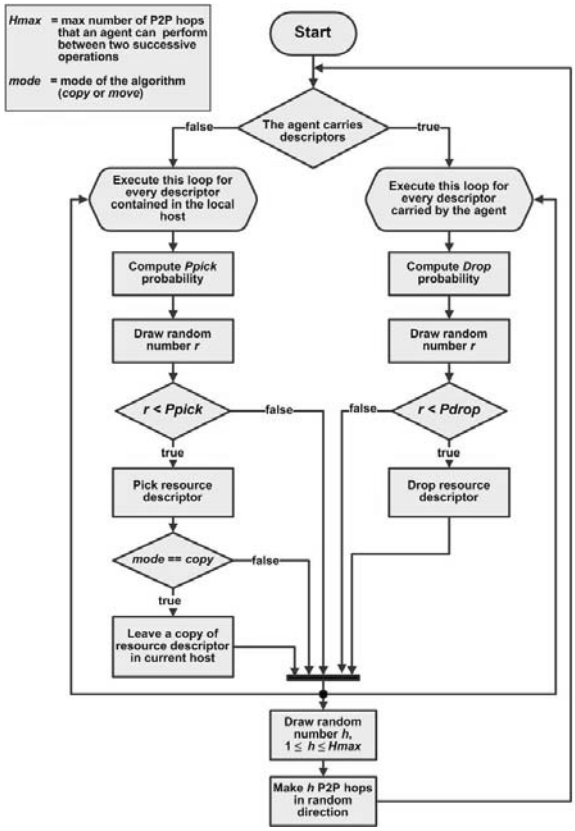

Fig. 1. Algorithm performed by ant-inspired agents.

III. The Antares Algorithm

The main purpose of the Antares algorithm is to disseminate resource descriptors over the Grid and at the same time achieve a logical organization of grid resources by spatially sorting the corresponding descriptors according to the their indexes.

The Antares information system is progressively and continuously constructed by a number of ant-inspired agents, which travel the grid through P2P interconnections and copy and reorganize descriptors. Though agent operations are very simple, and agents are unaware of the significance of what they do, a form of *swarm intelligence* emerges from their combined work, which is typical of ant systems and of many biological systems in general.

A high-level description of the algorithm is given in the flowchart of Fig. 1. Whenever an agent arrives at a new Grid host, it operates as follows: 1) if the agent does not carry any descriptor, it evaluates the *pick probability function* for every resource descriptor stored in this host, so as to decide whether or not to pick these descriptors; 2) if the agent already carries some descriptors, it evaluates the *drop probability function* for each of these descriptors, so as to decide whether or not to leave them in the current host. After picking descriptors, the agent will carry them until it drops them into another host, and then will try to pick other descriptors from another host. An agent can replicate descriptors, or simply move them, depending on the "mode" with which it operates, the *copy* or *move* mode. This will be better discussed in the following.

The *pick* and *drop* operations are driven by the corresponding probability functions that are defined and discussed in Sections III-A and III-B. Subsequently, Section III-C introduces the self-organizing mechanism that allows agents to switch from the replication (*copy*) to the relocation (*move*) modality. Finally, Section III-D discusses the mechanisms used to tackle the dynamic nature of the Grid and achieve a continuous turn-over of agents.

A. Pick Operation

Whenever an agent arrives at a new grid host, and it does not carry any descriptor, it evaluates the *pick probability function* and decides whether or not to pick one or more descriptors from the current host. Specifically, the agent checks each descriptor maintained in the current host, and evaluates its *average similarity* with all the descriptors maintained by the hosts in the *visibility region*. The *visibility region* includes all the hosts that are located within the *visibility radius*, i.e., which are reachable from the current host with a given number of hops. This radius is an algorithm parameter, and is set here to 1, in order to limit the amount of information exchanged among hosts.

The aim of the *pick* operation is to take a descriptor from a peer if it is dissimilar to the other descriptors located in the visibility region, so that this descriptor can be moved by the agent and placed in another region beside other similar descriptors. As soon as the possible initial equilibrium is broken (descriptors having different keys begin to be accumulated in different grid regions), a further reorganization of descriptors is increasingly driven, because the probability of picking an "outlier" descriptor further increases.

To obtain this effect, the picking probability, as well as the dropping probability discussed later, is defined starting from the similarity function f reported in (1), and its value is inversely proportional to the value of f. The function f measures the average similarity of a given descriptor $\overline{d}$ with all the other descriptors d located in the visibility region R. In (1), N_d is the overall number of descriptors maintained in the region R, while $H(d, \bar{d})$ is the Hamming distance between d and $\overline{d}$. The parameter α defines the similarity scale [15]; here it is set to $B/2$, which is half the value of the maximum Hamming distance between vectors having B bits. The value of f assumes values ranging between -1 and 1, but negative values are truncated to 0.

The pick probability function, P_{pick}, is given in (2). In this formula, the parameter k_p, whose value lies between 0 and 1, can be tuned to modulate the degree of similarity among descriptors. In fact, the pick probability is equal to 0.25 when f and k_p are comparable, while it approaches 1 when f is much lower than k_p (i.e., when $\overline{d}$ is extremely dissimilar from the other descriptors) and 0 when f is much larger than k_p (i.e., when $\overline{d}$ is very similar to others descriptors). In this paper, k_p is set to 0.1, as in [10]

$$f(\bar{d}, R) = \frac{1}{N_d} \cdot \sum_{d \epsilon R} \left(1 - \frac{H(d, \bar{d})}{\alpha}\right) \tag{1}$$

$$P_{\text{pick}} = \left(\frac{k_p}{k_p + f}\right)^2. \tag{2}$$

After evaluating the pick probability function, the agent computes a random real number between 0 and 1, and then it executes the pick operation if this number is lower than the value of the pick function. As the local region accumulates descriptors having similar keys, it becomes more and more likely that "outlier" descriptors will be picked by an agent.

The *pick* operation can be performed in two different modes, *copy* and *move*. If the *copy* mode is used, the agent, when executing a pick operation, leaves the descriptor on the current host, *generates a replica* of it, and carries the new descriptor until it drops it into another host. Conversely, with the *move* mode, an agent picks the descriptor and *removes* it from the current host, thus preventing an excessive proliferation of replicas. The use of these two modes is better discussed in Section III-C.

B. Drop Operation

The *drop probability function* P_{drop} is evaluated by an agent when it carries some previously picked descriptors, arrives at a new grid host, and must decide whether or not to deposit these descriptors. As with the pick function, it is first used to break the initial equilibrium and then to strengthen the spatial sorting of descriptors.

For each carried descriptor, the agent separately evaluates the drop probability function that, as opposed to the pick function, is directly proportional to the similarity function f defined in (1), i.e., to the average similarity of this descriptor with the descriptors maintained in the visibility region.

The P_{drop} is given in (3). The parameter k_d is set to a higher value than k_p, specifically to 0.5, in order to limit the frequency of drop operations. Indeed, it was observed that if the drop probability function tends to be too high, it is difficult for an agent to carry a descriptor for an amount of time sufficient to move it into an appropriate grid region.

As for the pick operation, the agent first evaluates P_{drop}, then extracts a random real number between 0 and 1, and, if the latter number is lower than P_{drop}, drops the descriptor in question into the current host

$$P_{\text{drop}} = \left(\frac{f}{k_d + f}\right)^2. \tag{3}$$

As a final remark concerning pick and drop probability functions, it is worth specifying that the values of mentioned parameters (k_p, k_d, α) have an impact on the velocity and duration of the transient phase of the Antares process, but they have little influence on the performance observed under steady conditions. In other words, Antares was found to be robust with respect to the variation of these parameters.

C. Replication and Relocation of Descriptors

The effectiveness of Antares is evaluated through the spatial *homogeneity function* H. Specifically, for each peer p of the grid, the average homogeneity H_p of the descriptors located in the visibility region of p, R_p, is calculated. This is obtained, as shown in (4), by averaging the Hamming distance between

every couple of descriptors in R_p and then subtracting the obtained value from B, which is equal to the maximum Hamming distance. Thereafter, the value of H_p is averaged over the whole Grid, as formalized in (5)

$$H_p = B - AVG_{\{d_1,d_2 \epsilon R_p\}}(H(d_1, d_2)) \tag{4}$$

$$H = \frac{1}{N_p} \cdot \sum_{p \epsilon \text{grid}} H_p. \tag{5}$$

The objective of Antares is to increase the homogeneity function as much as possible, because it would mean that similar descriptors are actually mapped and aggregated into neighbor hosts, and therefore an effective sorting of descriptors is achieved.

Simulation analysis showed that the homogeneity function is better increased if each agent works in both its operational modes, *copy* and *move*. In the first phase of its life, an agent is required to *copy* the descriptors that it picks from a grid host, but when it realizes from its own activeness that the sorting process is at an advanced stage, it begins simply to *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 store a very large number of descriptors of all types, thus weakening the efficacy of spatial reorganization. The algorithm is effective only if each agent, after replicating a number of descriptors, switches from *copy* to *move*.

A self-organization approach, which in some sense is similar to that used in [39], enables each agent to tune its activeness, in our case to perform a mode switch, only on the basis of local information. Our approach is inspired by the observation that agents perform more operations when the system entropy is high, because descriptors are distributed randomly, 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, which are defined in (2) and (3), decrease as descriptors are correctly replaced and sorted on the grid.

With a mechanism inspired by ants and other insects, 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 T_h. In particular, at given time intervals, specifically every 2000 s,[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 $N_{\text{operations}}$. At the end of each time interval, the agent makes a deposit into its pheromone base, which is inversely proportional to the fraction of performed operations. An evaporation mechanism is used to give a greater weight to the recent behavior of the agent. Specifically, at the end of the i-th time interval, the pheromone level Φ_i is computed with (6) and (7)

$$\Phi_i = Ev \cdot \Phi_{i-1} + \phi_i \tag{6}$$

$$\phi_i = 1 - \frac{N_{\text{operations}}}{N_{\text{attempts}}}. \tag{7}$$

The evaporation rate E_v is set to 0.9 [39], whereas ϕ_i is the amount of pheromone deposited in the last time interval. The pheromone level can assume values between 0 and 10: the superior 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 T_h (whose value must also be set between 0 and 10), the agent switches its mode from *copy* to *move*. The value of T_h can be used to tune the fraction of agents that work in the *copy* mode, and consequently the replication and dissemination of descriptors, since these agents are able to generate new descriptor replicas. Specifically, the number of agents in *copy* increases with the value of the threshold T_h.

D. Management of Peer Disconnections

In a dynamic grid, peers can go down and reconnect again with varying frequencies. To account for this, the *average connection time* of each peer is generated according to a Gamma probability function, with an overall 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.

As a consequence of this dynamic nature, two issues are to 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 remove "obsolete descriptors," i.e., descriptors of resources provided by hosts that have left the system, and therefore are no longer available.

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 life-time of these new agents to the average connection time of the peer itself. This setting ensures that 1) a proper turnover of agents is achieved, because old agents die when their lifetimes expire and new agents are generated by reconnecting peers and 2) the relation between the number of peers and the number of agents is maintained with time: more specifically, the overall number of agents is approximately equal to the number of active peers multiplied by N_{gen}. The agent turnover allows for the dissemination of descriptors of new resources, since new agents start in the *copy* mode.

A second mechanism ensures that, every time a peer disconnects from the grid, it loses all the descriptors previously deposited by agents, thus contributing to the removal of

[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 s value allows on average 33.3 operations to be aggregated, since the average interval between two agent movements is set to 60 s.

obsolete descriptors. Finally, a *soft state* mechanism [40] is adopted to avoid the accumulation of obsolete descriptors in very stable nodes. To keep information up to date, the soft state paradigm uses refresh messages that are sent periodically from the information source to the hosts that maintain the replicas. In Antares, information refresh is performed as follows. Each host checks the descriptors stored locally and starts the refresh procedure for the descriptors that have been deposited by agents since a given period of time. For these descriptors, the host contacts the peers that publish the related resources and requests refresh messages from them containing updated information about the state of those resources. The refresh interval and the minimum age of resources that must be updated should be set depending on the load of refresh messages that can be tolerated and the frequency at which resources can change their state.

It is worth mentioning that the described 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 1) a peer does not have to perform any procedure before leaving the system, and 2) in both cases (disconnection and fault) the descriptors that the peer has accumulated so far are removed.

IV. Evaluation of Dissemination and Spatial Sorting

The performance of the Antares algorithm was evaluated with an event-based simulator written in Java. Simulation objects are used to emulate grid peers and Antares 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 to be delivered to other objects.[2]

A Grid network having a number of hosts N_p equal to 2500 is considered in this paper. Hosts are linked through P2P interconnections, and each host is connected to four peer hosts on average. The topology of the network was built using the well-known scale-free algorithm defined by Albert and Barabasi [41], which 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 average connection time of a peer, T_{peer} (see Section III-D), is set to 100 000 s. 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.

Resources are characterized by metadata descriptors indexed by bit vectors (keys) having B bits, with $2^B - 1$ possible values. These values, as mentioned in Section I, can result from a semantic description of a resource, in which each bit represents the presence of a specific topic, or from the application of a locality preserving hash function.[3] In both cases, it is guaranteed that similar keys are given to descriptors of similar resources.

The mean number of agents generated by a single peer N_{gen} is set to 0.5; as a consequence, the average number of agents N_a that travel the Grid is approximately equal to $N_p/2$, as discussed in Section III-D. The average time T_{mov} between two successive agent movements is set to 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. It is assumed that the average link delay, i.e., the time needed by an agent to perform a hop, is 100 ms, with a Gamma probability distribution.

A set of performance indices are defined to evaluate the performance of the Antares algorithm. The overall homogeneity function H, which is defined in Section III-C, is used to estimate the effectiveness of the algorithm in the reorganization of descriptors. The N_d index is defined as the mean number of descriptors maintained by a grid host. Since new descriptors are only generated by agents that work in the *copy* mode (see Section III-C), the number of such agents N_{copy} is another interesting index that helps to understand what happens in the system. Finally, the processing load and the traffic load are also computed.

Performance indices have been obtained by varying several parameters, for example, the average number of resources published by a host and the frequency of agent movements. It was found that the qualitative behavior of Antares is not affected by these parameters, which confirms its robustness. Furthermore, Antares is scalable: since each agent operates only on the basis of local information, performance is not significantly affected by the size of the network, as shown in Section IV-A. Robustness and scalability derive from the decentralized, self-organizing, and adaptive features of Antares, and in general of the swarm intelligence paradigm.

In this paper, we choose to show performance indices versus time, obtained when varying two important parameters: T_{peer}, the average connection time of a peer, which has been introduced in Section III-D, and B, the number of bits of descriptor indices. Corresponding results are described in Sections IV-A and IV-B, respectively. The performance evaluation of simple and range queries will be discussed in Section V.

A. Performance Versus the Average Peer Connection Time

In the first set of tests, the value of B was set to 4, while tested values of T_{peer} were varied from 35 000 to 1 000 000 s. For comparison purposes, the case in which peers never disconnect was also tested. This kind of analysis is useful because it helps to understand the mechanisms through which the information system is constructed, and also because it is possible to assess the algorithm ability to adapt the mapping of descriptors to the continuous modifications of the environment. Simulations were executed with T_h equal to 9.0, which means that each agent sets off in the *copy* mode and passes to the

[2]This simulator is available on line at *http://antares.icar.cnr.it*. The reader can set network and protocol parameters, run his/her own simulations, and view the results. The simulator also allows saving and comparing results obtained in different scenarios, even dissimilar to those analyzed in this paper.

[3]A key with all bits equal to 0 is not permitted to make the approach consistent with both these definitions.

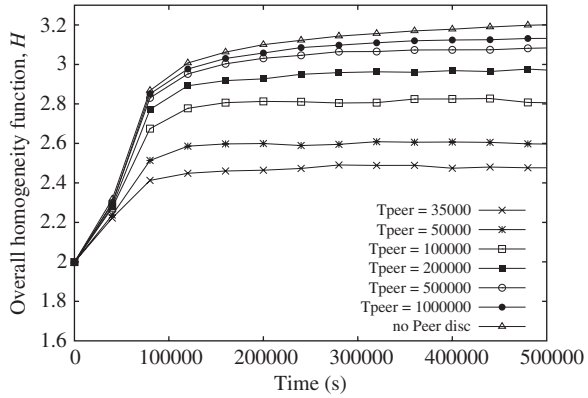

Fig. 2. Overall homogeneity function, vs. time for different values of the average connection time T_{peer}.

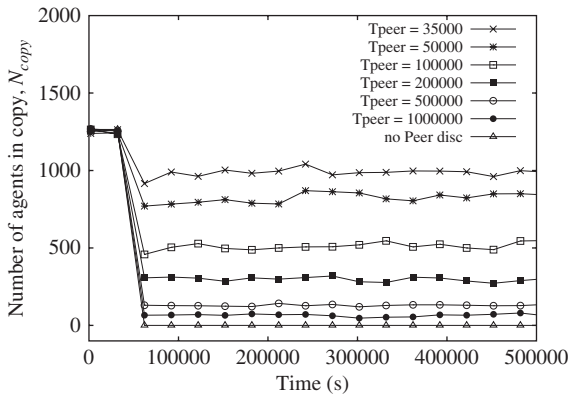

Fig. 3. Number of agents that operate in the *copy* mode vs. time for different values of the average connection time T_{peer}.

move mode as soon as its pheromone, starting from 0, exceeds the threshold of 9.0 (see Section III-C).

Fig. 2 reports the trend of H, which is the overall homogeneity function, assuming that the Antares process starts at time 0. The descriptors are initially distributed in a completely random fashion, but subsequently they are reorganized and spatially sorted by agents. It appears that the work of Antares agents make the H index increase from about 2 (i.e., $B/2$) to much higher values. After a transient phase, the value of H 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, etc. 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 H decreases as the network becomes more dynamic, with lower values of T_{peer}, because the reorganization of descriptors performed by agents is partly hindered by environmental modifications. However, even with the lowest value of T_{peer} that we tested, i.e., 35 000, the increase of H is about 25%, whereas it is more than 55% with T_{peer} equal to 1 000 000.

Fig. 3 depicts N_{copy}, which is the number of agents that operate in the *copy* mode, also called *copy agents* in the following. This analysis is interesting because these agents are responsible for the replication of descriptors, whereas agents in the *move* mode are exclusively devoted to the relocation and spatial sorting of descriptors. 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 T_h. This corresponds to the sudden drop of curves that can be observed in the left part of Fig. 3. Thereafter, the number of *copy* agents becomes stabilized, even with some fluctuations; this equilibrium is reached because the number of new agents that are generated by grid hosts (these agents always set off in the *copy* mode) and the number of agents that switch from *copy* to *move* become balanced.

Fig. 3 also shows that the number of *copy* agents increases as the grid is more dynamic. Indeed, a higher turnover of agents is obtained when peers disconnect and reconnect with a higher frequency, because more agents die if more peers disconnect (indeed, the lifetime of agents is correlated to the lifetime of peers), and at the same time more agents are generated by reconnecting peers (see Section III-D). Since new agents set off in the *copy* mode, this leads to a larger number of *copy* agents, as appears in Fig. 3. This is also due to another reason: as the rate of peer disconnections increases, the reorganization process is partly hindered, pick and drop functions assume lower values, and therefore agent operations become more frequent. The result is that the pheromone value increases more slowly, as evident in (6) and (7), and fewer agents switch to the *move* mode. Note that in a static network with no peer disconnections, all agents work in the *move* mode after a short transient phase. Indeed, in this case no new agents are generated after the process is initiated.

Fig. 4 reports N_d, which is the average number of descriptors that are maintained by a grid host at a given time. One of the main objectives of Antares is the replication and dissemination of descriptors. This objective is achieved; in fact, the value of N_d increases from an initial value of about 15 (equal to the average number of resources published by a host) to much higher values. As for the other indices, the trend of N_d undergoes a transient phase, and then it becomes stabilized, even though with some fluctuations.

The value of N_d is determined by two main phenomena: on the one hand, a large number of *copy* agents tend to increase N_d, because they are responsible for the generation of new descriptor replicas. On the other hand, a frequent disconnection of peers tends to lower N_d, because a disconnecting peer loses all the descriptors that it has accumulated so far, see Section III-D. These two phenomena work in opposite directions as the value of T_{peer} decreases; in a more dynamic network there are more *copy* agents, which tends to increase N_d, but more descriptors are thrown away by disconnecting peers, which tends to decrease N_d.

The result is that the stable number of N_d is relatively small both when the peer connection time is very low and when it is very high or infinite, i.e., with no peer disconnections. Interestingly, a higher degree of replication can be reached for intermediate values of T_{peer}, which are more realistic on grids. Indeed, the figure shows that even if curves are more wrinkled than those examined so far (probably due to the two underlying and contrasting mechanisms discussed above), the

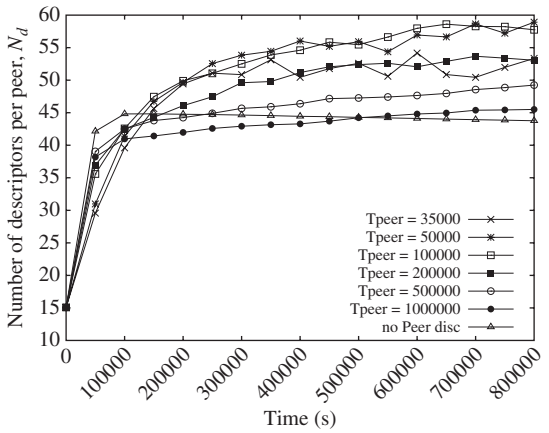

Fig. 4. Average number of descriptors maintained by a grid host vs. time for different values of the average connection time T_{peer}.

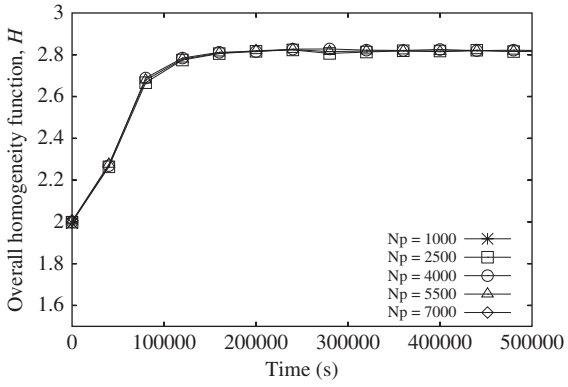

Fig. 5. Overall homogeneity function vs. time for different values of the number of peers N_p.

value of N_d first increases as T_{peer} increases from 35 000 to about 100 000, and then it decreases again for higher values of T_{peer}.

The scalability properties of Antares were analyzed by varying the number of peers N_p from 1000 to 7000. Interestingly, the size of the network has no detectable effect on the performance, specifically on the overall homogeneity index. This is observed in Fig. 5, in which the curves corresponding to the different values of N_p are almost completely overlapped. The scalable nature of Antares, which derives from its decentralized and self-organizing characteristics, is therefore confirmed. Similar considerations can be made for other performance indices, which are not reported here.

The processing load L_p is defined as the average number of agents per second that arrive at a node at the end of a complete movement (which can include several P2P hops) and are processed by a peer. It does not depend either on the churn rate or on the network size, but only on the average number of agents generated by a reconnecting peer N_{gen} and on the frequency of their movements across the grid $1/T_{\text{mov}}$. L_p is calculated by multiplying the overall number of agents N_a by the frequency of their movements $1/T_{\text{mov}}$, thereby obtaining the number of times per second that an agent arrives at any peer, and then dividing the result by the number of peers N_p,

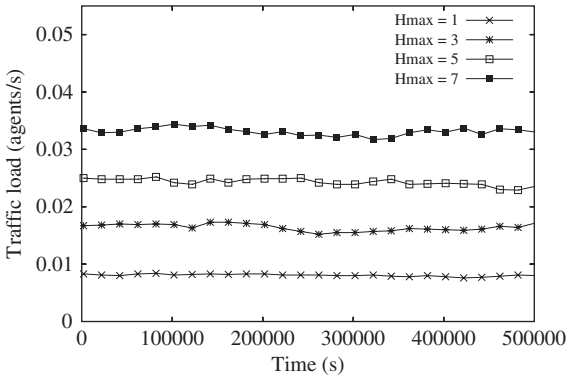

Fig. 6. Network traffic vs. time for different values of the parameter $H_{\max}$.

thus obtaining the number of times per second that an agent arrives at a specific peer

$$L_p = \frac{N_a}{N_p \cdot T_{\text{mov}}} \approx \frac{N_{\text{gen}}}{T_{\text{mov}}}. \tag{8}$$

In the described scenario, since the average value of T_{mov} is equal to 60 s, and N_{gen} is set to 0.5, each peer receives and processes about one agent every 120 s, which can be considered an acceptable load. This theoretical result has been confirmed by simulation data. Note that the processing load does not depend on other system parameters such as the network size, the average number of resources published by a node, and so on, which confirms the scalability properties of the Antares algorithm.

Finally, the traffic load is defined as the number of agents that go through a node per unit time. As shown in Fig. 6, the traffic load increases with the value of $H_{\max}$, which is the maximum number of P2P hops that are performed within a single agent movement. It was also found, however, that the reorganization of descriptors is accelerated if agent movements are longer, because they can explore the network more quickly. Therefore, the choice of $H_{\max}$ depends on a compromise between the desired efficiency of the reorganization process and the traffic load that can be tolerated.

B. Performance Versus the Granularity of Resource Classification

In the second set of simulation tests, the value of T_{peer} was set to 100 000 s, and the value of B was varied from 3 to 6. Since the number of different descriptor keys is equal to $2^B - 1$, this analysis allows us to understand how the reorganization and replication process can be affected by the different levels of detail through which resources are described and distinguished from each other. As in previous simulations, the pheromone threshold T_h was set to 9.0.

Fig. 7 reports the values of H, which is the overall homogeneity function discussed in Section III-C. It confirms that the work of Antares agents makes this index increase from about $B/2$ to much higher values. It is interesting to note that the increase in H hardly depends on the value of B, which means that the algorithm is able to reorganize descriptors regardless of the accuracy with which resources are described.

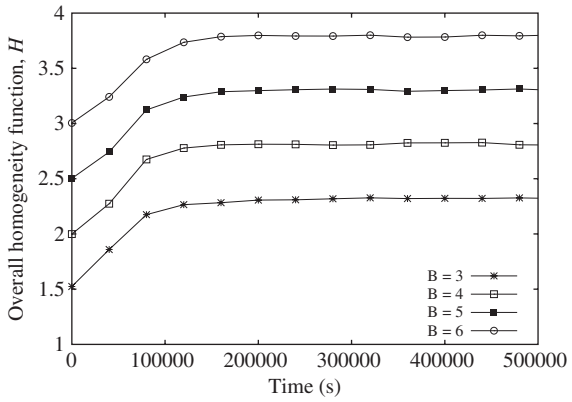

Fig. 7. Overall homogeneity function vs. time for different values of the number of bits in resource descriptors, B.

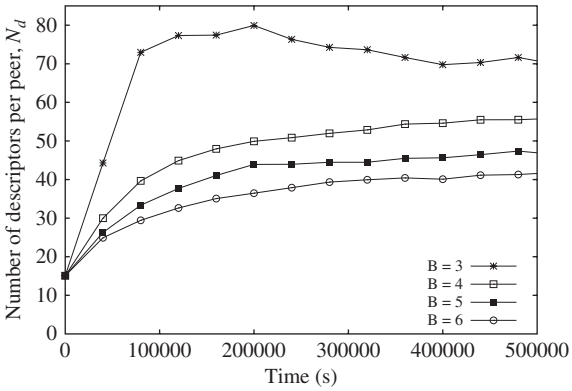

Fig. 8. Average number of descriptors maintained by a grid host vs. time for different values of the number of bits in resource descriptors B.

Fig. 8 reports N_d, which is the average number of descriptors that are maintained by a grid host, and shows that this index decreases as the value of B increases. The intuitive rational of this is the following: if the classification of resources is more refined, i.e., with a larger value of B, it becomes more difficult to discriminate among resources of different types, so pick and drop probability functions assume lower values and the replication process is attenuated. Conversely, with a lower value of B, an agent that carries a descriptor has more chances to find a region in which other descriptors having the same binary key have already been clustered.

This conjecture is confirmed by the results reported in Fig. 9, which depicts F_{pick} and F_{drop}, which are the number of pick and drop operations per second that are performed by all the agents. Indeed, more operations are performed for lower values of B. Another interesting outcome is that, for a fixed value of B, the frequencies of pick and drop operations converge to similar values, confirming that the system reaches an equilibrium despite all the underlying dynamic phenomena.

V. Resource Discovery Algorithm

In a distributed system, users often need to locate resources belonging to a given class (e.g., a computer or a web service with given characteristics) and, after retrieving a number of them, they can choose the resources that best fit their needs.

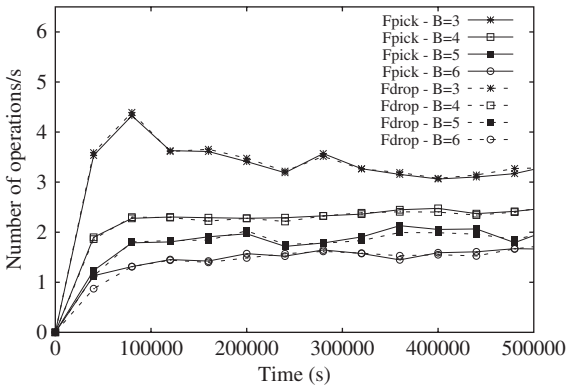

Fig. 9. Number of pick and drop operations performed per second by all the agents, for different values of the number of bits in resource descriptors B.

Accordingly, a query message is issued by a peer, on behalf of a user, to search for "target descriptors," that is, for resource descriptors having a given value of their binary index. The query must be forwarded through the network, hop by hop, so as to discover as many target descriptors as possible.

Thanks to the spatial sorting of descriptors achieved by Antares agents, the discovery procedure can be simply managed by forwarding the query, at each step, towards the "best neighbor," which is the neighbor peer that maximizes the similarity between the descriptors stored locally and the target descriptor. More specifically, each peer calculates a "centroid" descriptor. This descriptor is a vector of B real numbers lying between 0 and 1, and is obtained by averaging all the descriptor indexes of the local peer. The value of each centroid element is calculated by averaging the values of the bits, in the same position, of all the descriptors stored in the local peer. For example, the centroid descriptor of a peer that maintains the three descriptors [1, 0, 0], [1, 0, 0], and [0, 1, 0] is a descriptor having an index [0.67, 0.33, 0].

Before forwarding a query, a peer (say, peer p_A) calculates the cosine of the angle between the query target descriptor, and the centroids of all the neighbor peers. This value gives a hint about how much the descriptors of the neighbor peers are similar to the target descriptor. Thereafter, p_A forwards the query to the peer, say p_M, that maximizes this cosine similarity index. At the next step, p_M will do the same so that step by step the query approaches a region of the grid where it is more and more likely to discover several useful "results," that is, target descriptors. The search is terminated whenever it is no longer advantageous to forward the query, that is, when the best neighbor is no better than the peer where the query has arrived so far. At this point, a *queryHit* message is issued and returns to the requesting peer by following the same path, and collecting on its way all the results that it finds.

This discovery algorithm, even though very simple and demands very little computing and memory resources, is very efficient as it exploits the continuous work of mobile agents that reorganize descriptors on the grid. This is confirmed by Fig. 10, which depicts what can be called the "similarity improvement," obtained as the difference between the "arrival similarity" and the "departure similarity." These are the cosine similarities between the query target descriptor and the

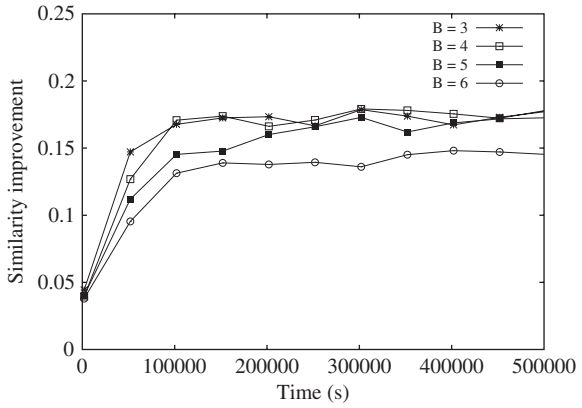

Fig. 10. Similarity improvement for different values of the number of bits in resource descriptors B.

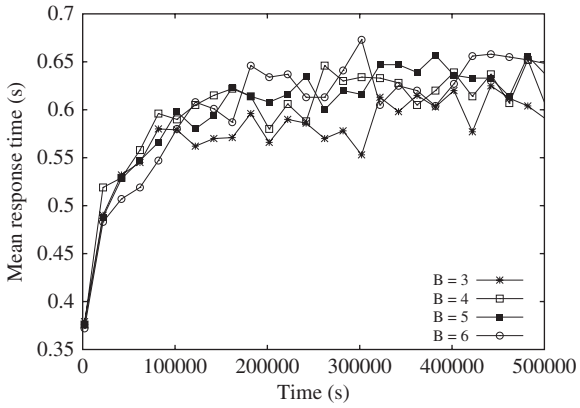

Fig. 12. Average response time of queries for different values of the number of bits in resource descriptors B.

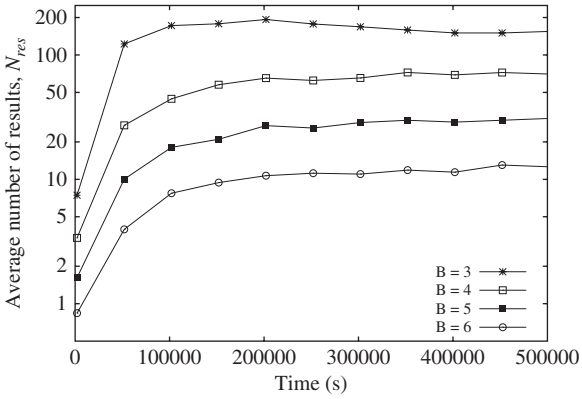

Fig. 11. Average number of results for different values of the number of bits in resource descriptors B. To improve the figure clearness, a log scale is used for the y-axis.

descriptor of the local centroid, respectively, calculated at the end and at the beginning of the query journey. Fig. 10 shows that the objective of the discovery algorithm, which of course is to increase the similarity difference as much as possible, is actually achieved. In fact, as the system evolves the cosine similarity is more than quadrupled with values of B equal to 3, 4, and 5, whereas the improvement is slightly lower with B equal to 6.

The ultimate objective of queries is to collect as many target descriptors as possible. The average number of results per query N_{res} is shown in Fig. 11. Thanks to the similarity improvement obtained by queries, the number of results remarkably increases with time, meaning that discovery operations are more and more efficient as descriptors are spatially sorted by agents. The number of results is obviously inversely proportional to B, since the fraction of target descriptors with respect to the totality of descriptors available on the network is on average equal to $1/(2^B - 1)$. In other words, a finer classification of resources corresponds to a lower probability of finding a target descriptor. Therefore, the value of B should be set on the basis of each particular application domain, taking into considerations that classification in larger classes can facilitate discovery, whereas, on the other hand, the definition of smaller classes can improve the *quality* of the discovered resources.

Finally, Fig. 12 shows the average response time, which is taken as the time interval between the query generation and the arrival of the queryHit message. It is assumed here that the time taken by a query or queryHit to perform a hop is 100 ms, with a Gamma probability distribution. The response time increases in the first phase, and then becomes stabilized at values between 600 and 650 ms, regardless of the value of B. The response time increase should not be seen as a negative effect (response times are in any case acceptable) but as a consequence of the fact that, with descriptor reorganization, queries can be successfully driven towards target descriptors for a larger number of hops.

A. Performance of Range Queries

So far, only simple queries, which are issued to find specific target descriptors, have been analyzed. However, as discussed in Section II, the efficient resolution of range queries is a fundamental requirement of grid information systems. In Antares, a range query is defined as a query in which the bit vector of the target descriptor contains one or more "wildcard" bits that can assume either 0 or 1. This means that a range query can return descriptors having 2^W possible values, if W:$W < B$ is the number of wildcard bits. Of course, the assignment of bit vectors to descriptors must ensure that the vectors that correspond to similar resources are also similar to one another. This can be done by using the binary Gray code, in which two successive indexes always differ by only one bit.

The discovery algorithm discussed in the previous section is slightly modified for the management of range queries. To select the best neighbor peer, the cosine similarity is still calculated between the target descriptor and the centroid descriptors of the neighbor peers, but this time these indexes are preprocessed by discarding the bits that are defined as wildcards in the target descriptor. Therefore, only the centroid bits that correspond to valued bits in the target descriptor are useful to drive the query message. As for simple queries, a range query terminates its journey when it is no more possible to find a better neighbor. The queryHit message will come back and collect all the descriptors that match the range query.

To evaluate the effectiveness of range queries, B was fixed to 4 and peers were made to issue three kinds of queries:

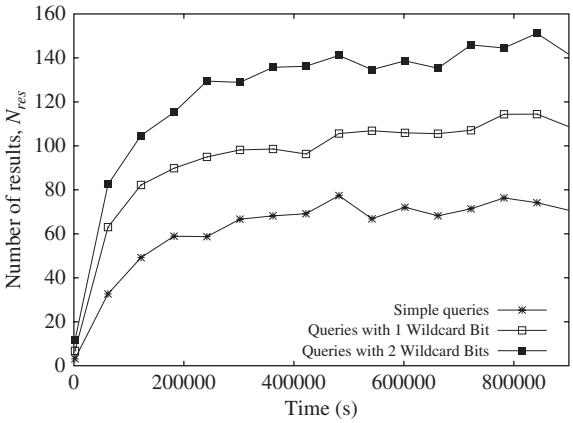

Fig. 13. Average number of results of range queries with $B = 4$ and different numbers of wildcard bits.

simple queries; range queries with 1 wildcard bit, chosen randomly; and range queries with 2 wildcard bits, also chosen randomly. The corresponding average number of results is shown in Fig. 13. It appears that, in a steady situation, a range query with 1 wildcard bit discovers about 110 results, that is 53% more results than a simple query, which discovers about 75 results on average. Two simple queries (precisely, the queries in which the wildcard bit of the target descriptor is respectively set to 0 and to 1, and all the other bits are not varied) would together discover more results, 150 on average, but at the cost of doubling the whole discovery procedure, thus increasing the processing and traffic load. Similarly, a range query with two wildcard bits is able to discover 106% more results than a simple query. Thus, range queries are not able to discover all the results that would be found with the corresponding number of simple queries (which is equal to 2^W), but provide an efficient way to discover many more results than a simple query in just one shot, which is the actual objective of a range query. It can be concluded that the resource management of Antares facilitates this objective.

B. Future Enhancements of Antares

It must be remarked that the discovery algorithm, especially for range queries, has been kept very simple in order to assess the effectiveness of resource reorganization and to fairly compare simple and range queries. However, several enhancements can be adopted to increase the number of results returned by discovery operations. A first enhancement could aim to overcome the problem of local maxima, which in some cases can limit the effectiveness of discovery operations. This can be done by enabling "long hops" of queries when they are still far from the target descriptors. This way, a query could overcome a local maximum and approximate the objective more quickly. The parallel issue of several query messages can further improve the query performance, though at the cost of increasing the network traffic. The long hop strategy could be adopted for only a fraction of these messages. A second enhancement could be a better investigation of an interesting region. For example, when a query terminates its journey, the queryHit message could explore all the neighbor peers and collect other useful results there, before initiating the return journey. In fact, owing to the gradual sorting of descriptors, these neighbor peers are likely to maintain a consistent number of target descriptors. This technique could be particularly useful in the case of range queries since target descriptors are generally spread in a larger region with respect to simple queries. These two enhancements are currently under examination.

Another interesting avenue for future research is the extension of Antares to the reorganization and discovery of multi-attribute resources. If it is assumed that a resource is characterized by M independent attributes (for example a computer could be characterized by its CPU rate and memory size), and each attribute can be mapped to a specific key through a locality preserving hash function. Therefore, the descriptor of a resource is associated with M keys, one for each attribute. The reorganization of descriptors can be performed by Antares agents in the same way as described in this paper, but descriptor keys related to different attributes are reorganized and sorted independently of each other. The descriptor of a resource can be accessed via any of its attributes, by searching the related key. A similar schema is adopted in the MAAN system [7], which enhances Chord in order to support the multi-attribute case. To serve a multiple attribute query, two strategies can be envisaged, like in [7], and are currently under examination.

1) With an *iterative approach*, a *subquery* is issued for every attribute of the multi-attribute query, and the final result is obtained as the intersection of the results of all the subqueries.
2) With the *dominant attribute* approach, only one subquery, which corresponds to the most selective attribute in the multi-attribute query, is issued. Each discovered descriptor that matches this subquery is immediately matched to the subqueries that correspond to the other attributes. Only the descriptors that satisfy all the subqueries are returned to the requester.

The latter strategy can considerably reduce the number of hops necessary to solve the multi-attribute query, but requires a larger computation load to process each candidate descriptor.

VI. Conclusion

In this paper, we introduced and evaluated Antares, which is an algorithm inspired by ant behavior and whose aim is to build a P2P information system of a grid. Through the evaluation of simple probability functions (*pick* and *drop*), a number of ant-inspired agents replicate and move the descriptors of grid resources from host to host, and this way disseminate and reorganize these descriptors on the network.

Antares achieves an effective reorganization of information, since descriptors are spatially sorted on the network and, in particular, descriptors indexed by equal or similar binary keys are placed in neighbor grid hosts. This was confirmed in the paper through the analysis of performance measures, in particular of a homogeneity index based on the Hamming distance between binary vectors. The reorganization of descriptors performed by Antares spontaneously adapts to the

ever changing environment, for example to the joins and departs of grid hosts and to the changing characteristics of resources. For these characteristics, the Antares approach has been named "self-structured."

We hope that this paper can open novel and promising avenues for the construction of distributed information systems and related discovery algorithms. In particular, Antares features two noteworthy properties, that are interesting enhancements with respect to most existing strategies: 1) it does not rely on any centralized support, but it is fully decentralized, self-organizing and scale-free, thanks to its bio-inspired nature and its swarm intelligence characteristics, and 2) being basically unstructured, it avoids the typical problems of structured systems, but still retains some of their important benefits, such as the efficient management of simple and range queries, thus proposing itself as a good compromise between the two strategies, structured and unstructured, which are generally deemed as complementary to each other.

ACKNOWLEDGMENT

We would like to thank G. Spezzano for his help in defining the bio-inspired algorithms presented in this paper.

REFERENCES

[1] I. Foster and C. Kesselman, *The Grid 2: Blueprint for a New Computing Infrastructure*. San Francisco, CA: Morgan Kaufmann, 2003.

[2] The Globus Alliance. (2006). The Web Services Resource Framework [Online]. Available: http://www.globus.org/wsrf/

[3] A. Iamnitchi, I. Foster, J. Weglarz, J. Nabrzyski, J. Schopf, and M. Stroinski, "A peer-to-peer approach to resource location in grid environments," in *Grid Resource Management*, Norwell, MA: Kluwer-Nijhoff, 2003.

[4] I. J. Taylor, *From P2P to Web Services and Grids: Peers in a Client/Server World*. New York: Springer-Verlag, 2004.

[5] A. Crespo and H. Garcia-Molina, "Routing indices for peer-to-peer systems," in *Proc. 22nd Int. Conf. Distributed Computing Syst. ICDCS '02*, Jul. 2002, pp. 23–33.

[6] C. Platzer and S. Dustdar, "A vector space search engine for web services," in *Proc. 3rd Eur. Conf. Web Services ECOWS 2005*, Washington, D.C.: IEEE Comput. Soc., 2005, pp. 62–71.

[7] M. Cai, M. Frank, J. Chen, and P. Szekely, "Maan: A multi-attribute addressable network for grid information services," *J. Grid Comput.*, vol. 2, no. 1, pp. 3–14, Mar. 2004.

[8] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat, "Design and implementation tradeoffs for wide-area resource discovery," in *Proc. 14th IEEE Int. Symp. High Performance*, Raleigh, NC: Research Triangle Park, Jul. 2005, pp. 113–124.

[9] A. Forestiero, C. Mastroianni, and G. Spezzano, "Antares: An ant-inspired P2P information system for a self-structured grid," in *Proc. 2nd Int. Conf. Bio-Inspired Models Network, Inform. Comput. Syst. Bionetics 2007*, Budapest, Hungary, Dec. 2007, pp. 151–158.

[10] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. New York: Oxford Univ. Press, 1999.

[11] J. M. Whitacre, R. A. Sarker, and Q. T. Pham, "The self-organization of interaction networks for nature-inspired optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 2, pp. 220–230, May 2007.

[12] M. Dorigo, E. Bonabeau, and G. Theraulaz, "Ant algorithms and stigmergy," *Future Generation Comput. Syst.*, vol. 16, no. 9, pp. 851–871, 2000.

[13] A. Forestiero, C. Mastroianni, and G. Spezzano, "So-grid: A self-organizing grid featuring bio-inspired algorithms," *ACM Trans. Autonomous Adaptive Syst.*, vol. 3, no. 2, May 2008.

[14] A. Forestiero, C. Mastroianni, and G. Spezzano, "Reorganization and discovery of grid information with epidemic tuning," *Future Generation Comput. Syst.*, vol. 24, no. 8, pp. 788–797, 2008.

[15] E. D. Lumer and B. Faieta, "Diversity and adaptation in populations of clustering ants," in *Proc. SAB94, 3rd Int. Conf. Simulation Adaptive Behavior: From Animals to Animats 3*, Cambridge, MA: MIT Press, 1994, pp. 501–508.

[16] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Comput. Surveys*, vol. 36, no. 4, pp. 335–371, 2004.

[17] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. Conf. Applicat., Technol., Architectures Protocols Comput. Commun. SIGCOMM '01*. New York: ACM, pp. 149–160.

[18] G. Sakaryan, M. Wulff, and H. Unger, "Search methods in P2P networks: A survey," in *Proc. Innovative Internet Community Syst., (IICS '04)*, LNCS vol. 3473/2006. Guadalajara, Mexico: Springer-Verlag, Mar. 2006.

[19] C. Mastroianni, D. Talia, and O. Verta, "Designing an information system for grids: Comparing hierarchical, decentralized p2p and super-peer models," *Parallel Comput.*, vol. 34, no. 10, pp. 593–611, Oct. 2008.

[20] M. Marzolla, M. Mordacchini, and S. Orlando, "Peer-to-peer systems for discovering resources in a dynamic grid," *Parallel Comput.*, vol. 33, no. 4–5, pp. 339–358, 2007.

[21] D. C. Erdil, M. J. Lewis, and N. Abu-Ghazaleh, "An adaptive approach to information dissemination in self-organizing grids," in *Proc. Int. Conf. Autonomic Autonomous Syst. ICAS '06*, Silicon Valley, CA, Jul. 2006, pp. 55–60.

[22] A. Padmanabhan, S. Wang, S. Ghosh, and R. Briggs, "A self-organized grouping (SOG) method for efficient grid resource discovery," in *Proc. 6th IEEE/ACM Int. Workshop Grid Comput.*, Seattle, WA, Nov. 2005, pp. 312–317.

[23] A. J. Chakravarti, G. Baumgartner, and M. Lauria, "The organic grid: Self-organizing computation on a peer-to-peer network," *IEEE Trans. Syst., Man, Cybern., Part A*, vol. 35, no. 3, pp. 373–384, May 2005.

[24] K. Sycara, "Multiagent systems," *Artificial Intell. Mag.*, vol. 10, no. 2, pp. 79–93, 1998.

[25] M. Fukuda and D. Smith, "UWAgents: A mobile agent system optimized for grid computing," in *Proc. 2006 Int. Conf. Grid Comput. Applicat.*, Las Vegas, NV, Jun. 2006, pp. 107–113.

[26] R. F. Lopes, F. J. da Silva, and B. B. de Sousa, "MAG: A mobile agent based computational grid platform," in *Proc. Int. Conf. Grid Cooperative Comput. (GCC '05)*, Beijing, China, Nov.–Dec. 2005, pp. 262–273.

[27] P. Grassé, "La reconstruction du nid et les coordinations inter-individuelles chez belicositermes natalensis et cubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs." *Insectes Sociaux*, no. 6, pp. 41–84, 1959.

[28] O. Babaoglu, H. Meling, and A. Montresor, "Anthill: A framework for the development of agent-based peer-to-peer systems," in *Proc. 22nd Int. Conf. Distributed Comput. Syst. ICDCS '02*, Washington, D.C.: IEEE Comput. Soc., pp. 15–22.

[29] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," in *Proc. Special Interest Group Data Commun. ACM SIGCOMM '02*, Pittsburgh, PA, pp. 177–190.

[30] A. Iamnitchi and I. Foster, "Interest-aware information dissemination in small-world communities," in *Proc. 14th IEEE Int. Symp. High Performance Distributed Comput., HPDC.*, Raleigh, NC: Research Triangle Park, Jul. 2005, pp. 167–175.

[31] M. S. Aktas, G. C. Fox, and M. Pierce, "Fault tolerant high performance information services for dynamic collections of grid and web services," *Future Generation Comput. Syst.*, vol. 23, no. 3, pp. 317–337, 2007.

[32] D. Tsoumakos and N. Roussopoulos, "A comparison of peer-to-peer search methods," in *Proc. 6th Int. Workshop Web Databases WebDB '03*, San Diego, CA, Jun. 2003, pp. 61–66.

[33] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Proc. 16th Int. Conf. Supercomput. ICS '02*, New York: ACM, Jun. 2002, pp. 84–95.

[34] D. Tsoumakos and N. Roussopoulos, "Adaptive probabilistic search for peer-to-peer networks," in *Proc. 3rd IEEE Int. Conf. P2P Comput. P2P '03*, Linköping, Sweden, Sep. 2003, pp. 102–109.

[35] A. S. Cheema, M. Muhammad, and I. Gupta, "Peer-to-peer discovery of computational resources for grid applications," in *Proc. 6th IEEE/ACM Int. Workshop Grid Comput.*, Seattle, WA, 2005, pp. 179–185.

[36] A. Gupta, D. Agrawal, and A. El Abbadi, "Approximate range selection queries in peer-to-peer systems," in *Proc. 1st Biennial Conf. Innovative Data Syst. Research*, Asilomar, CA, 2003.

[37] A. Andrzejak and Z. Xu, "Scalable, efficient range queries for grid information services," in *Proc. 2nd IEEE Int. Conf. Peer-to-Peer Computing P2P '02*, Washington, D.C.: IEEE Comput. Soc., pp. 33–40.

[38] C. Schmidt and M. Parashar, "Enabling flexible queries with guarantees in p2p systems," *IEEE Internet Comput.*, vol. 8, no. 3, pp. 19–26, May–Jun. 2004.
[39] H. V. D. Parunak, S. Brueckner, R. S. Matthews, and J. A. Sauter, "Pheromone learning for self-organizing agents," *IEEE Trans. Syst., Man Cybern., Part A*, vol. 35, no. 3, pp. 316–326, May 2005.
[40] P. Sharma, D. Estrin, S. Floyd, and V. Jacobson, "Scalable timers for soft state protocols," in *Proc. 16th Annu. Joint Conf. IEEE Comput. Commun. Soc., INFOCOM '97*, vol. 1. Washington, DC: IEEE Comput. Soc., pp. 222–229.
[41] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Sci.*, vol. 286, no. 5439, pp. 509–512, Oct. 1999.

Agostino Forestiero received the Laurea degree in computer engineering and the Ph.D. degree in computer engineering from the University of Calabria, Cosenza, Italy, in 2002 and 2007, respectively.

He is a currently Research Fellow at the CNR Institute for High Performance Computing and Networks, Rende, Italy. He has published or presented more than 30 scientific papers on international journals and conferences. His research interests include grid computing, peer-to-peer networks, and swarm intelligence.

Dr. Forestiero is a member of the IEEE Computer Society. He has served as a Program Committee Member of several conferences.

Carlo Mastroianni received the Ph.D. in computer engineering from the University of Calabria, Cosenza, Italy, in 1999.

He has been a Researcher with the CNR Institute for High Performance Computing and Networks, Rende, Italy, since 2002. His research interests focus on distributed systems and networks, and in particular grid computing, peer-to-peer networks, content distribution networks, and multiagent systems. He has published or presented more than 80 scientific papers in international journals and conferences.

Dr. Mastroianni is a member of the ACM and the IEEE Computer Society. He has served as Chair or Program Committee Member of several conferences.