Utility-based Mechanism for Structural Self-Organization in Service-Oriented MAS

E. del Val, Universitat Politècnica de València
M. Vasirani, Ècole Polytechnique Fédérale de Lausanne
M. Rebollo, Universitat Politècnica de València
A. Fernández, Universidad Rey Juan Carlos

The structural relations established among agents influence the performance of decentralized service discovery process in Multi-Agent Systems (MAS). Moreover, distributed systems should be able to adapt their structural relations to changes in environmental conditions. In this paper, we present a Service-Oriented MAS where agents initially self-organize their structural relations based on the similarity of their services. During the service discovery process, agents integrate a mechanism that facilitates the self-organization of their structural relations in order to adapt the structure of the system to the service demand. This mechanism facilitates the task of decentralized service discovery and improves its performance. Each agent has local knowledge about its direct neighbors and the queries received during discovery processes. With this information, an agent is able to analyze its structural relations and decide when it is more appropriate to modify its direct neighbors and select the most suitable acquaintances to replace them. The experimental evaluation shows how this self-organization mechanism improves the overall performance of the service discovery process in the system when the service demand changes.

Categories and Subject Descriptors: I.2 [**Computing Methodologies**]: Artificial Intelligence

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Self-Organization, Service Discovery, Complex Networks, Dynamic Networks

ACM Reference Format:
E. del Val, M. Vasirani, M. Rebollo, A. Fernández, 2014. Utility-based Mechanism for Structural Self-Organization in Service-Oriented MAS. *ACM Trans. Embedd. Comput. Syst.* 9, 4, Article 39 (March 2010), 25 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Nowadays, there is a trend towards large-scale, complex, and highly-dynamic systems for dealing with new business models and requirements [Werbach 2000]. Service-Oriented Multi-Agent Systems (SOMAS) are considered to be a technology that supports these new models when there is a large number of entities offering services that change frequently and that look for other entities to collaborate with in order to obtain a resource to deal with a complex goal [Huhns 2002; Huhns and et al. 2005; Brazier et al. 2009]. SOMAS integrate Service-Oriented Computing (SOC) and Multi-Agent System (MAS) technologies where: (i) service standards provide an infrastructure for

This work is partially supported by the Spanish Ministry of Science and Innovation through grants CSD2007-0022 (CONSOLIDER-INGENIO 2010), TIN2012-36586-C03-01, TIN2012-36586-C03-01, TIN2012-36586-C03-02, and FPU grant AP-2008-00601 awarded to E. del Val.

DOI: http://dx.doi.org/10.1145/0000000.0000000

the interaction among agents; (ii) MAS offer a more general and complex notion of Service-Oriented Architectures (SOA); and (iii) intelligent and social capabilities of agents allow complex systems to be defined.

In SOMAS, services are considered to be the basic building blocks of complex business applications. Services are platform-independent and can be described, discovered, and composed dynamically. These features make services suitable for giving support to the high rate of change in business demands. However, in order to provide more flexibility in the context of business applications, services should be both reactive and proactive. They should be aware of what is happening in their environment and also be able to perform local actions based on their observations. Agents are able to learn from previous experiences and update and reason about their information in order to improve their decisions and achieve their goals. Moreover, SOMAS should provide mechanisms to provide higher levels of functionality and to facilitate the emergence of new services in a dynamic way by exploiting existing services.

Service discovery is a challenging task for SOMAS when changes in the environment occur (i.e., distribution of service demand, agents that leave and enter the system) and there is no central repository responsible for the management of resources and the maintenance of the system structure. Therefore, each agent should be able to locate another agent that provides the required service and to update its structural links to obtain more useful relations. The success of the service discovery process relies on the collaboration of other agents in the system [Del Val et al. 2012b] and the self-organization of the structural relations between agents [Abdallah and Lesser. 2007; Gaston and desJardins 2005; Kota et al. 2012].

In systems where the environmental conditions or requirements change and nodes only have local knowledge, the inclusion of self-organization mechanisms offers advantages such as increased scalability and robustness and a reduced need for communication. In this paper, we present a decentralized service discovery system that integrates a mechanism to facilitate the self-organization of the structural relations established among agents in order to adapt the system structure to the service demand. The self-organization mechanism considers local knowledge about interactions with direct neighbors during the discovery process. With this information, each agent is able to reason about when it is more appropriate to modify its structural relations with its direct neighbors and determine which acquaintances are the most suitable to replace them. Some of the scenarios where the proposal presented here can be applied are: file sharing P2P systems [Sun and Garcia-Molina 2004], streaming applications [Lin et al. 2009], overlay routing [Blanc et al. 2005], network of services [ITAO et al. 2001; Viroli and Zambonelli 2010], and sensor networks [Fernandez-Marquez et al. 2012] among others.

The rest of the paper is organized as follows: Section 2 presents a review of related work about decentralized search of resources and self-organization proposals. Section 3 describes the context where the proposed service discovery process and self-organization mechanism are going to be applied. In Section 4, we present our formal model for decentralized service discovery which underpins our proposed self-organizing mechanism. Section 5 explains the service discovery process and the self-organization of the structural relations between agents. A set of experiments to validate the structural self-organization mechanisms are presented in Section 6. Finally, some conclusions and final remarks are presented in Section 7.

2. RELATED WORK

Nowadays, decentralized systems appear as an alternative to traditional centralized approaches. The evolution of the Internet and communication, and the emergence of new market models have generated new requirements such as decentralized resource

search or dynamic self-organization for changes in the environment in order to improve system performance. In large-scale systems where there is a lack of global knowledge, decentralized search and dynamic self-organization generate new challenges such as dealing with uncertainty or action coordination based only on local states. These challenges cannot be tackled by traditional approaches [(ed.) di Marzo Serugendo et al. 2011; Biskupski et al. 2007]. In this section, we present several works that deal with decentralized service discovery and self-organization in distributed systems.

2.1. Search in Unstructured Environments

Search approaches that are commonly used in decentralized systems (where all the entities are considered to be equal and there is an arbitrary topology) are based on *blind* or *informed* algorithms. *Blind* algorithms do not consider any information about resource locations, and they use *flooding* or *random* strategies that can overload the system with the traffic generated during the search process [Ouksel et al. 2004; Zhong 2006]. To prevent the generation of traffic, *informed* algorithms that take into account local information have been proposed [Crespo and Garcia-Molina 2002; Basters and Klusch 2006]. The information is about their direct neighbors or statistics from previous searches which is stored in local registries. These algorithms require a period of time to collect information that can improve the search. If links between peers change frequently, statistical information that is stored in local indexes could become useless. Also, some of the heuristics that are used to guide the search process could overload some peers and leave other potential peers without traffic.

There are other informed approaches where the underlying structure of the system is *loosely structured* using certain criteria. This facilitates the search process [Zhang et al. 2004; Bianchini et al. 2009]. Initially, agents are connected randomly and they use a reorganization algorithm to group agents with similar services together. In order to avoid isolated clusters of agents, these algorithms establish a percentage of similar and dissimilar agents that are in the neighborhood of the agent. For distributed searches, agents use algorithms that are based on similarity; however, if they do not find any similar service, they use random algorithms. The main disadvantage of these approaches is the high cost of communication required to organize the entities into communities and the consideration of a fixed number of neighbors that are similar and dissimilar to the agent, which reduces the flexibility of the system.

Some of the approaches that use blind or informed strategies to locate resources in decentralized systems consider *Semantics*. We understand *Semantics* to be the introduction of machine interpretable languages in the descriptions of resources. Therefore, *Semantics* plays an important role in reducing the participation of the user in the service discovery process. Specifically, the inclusion of *Semantics* in the service discovery process implies the use of ontologies and semantic markup languages such as OWL-S [1], SAWSDL [2], or WSMO [3]. Semantic markup languages provide a formal and explicit specification of shared concepts. These languages facilitate the description of services and queries with a logic formalization. Markup languages exploit ontologies to facilitate sharing, reuse, composition, and mapping, which makes services computer-interpretable. As a consequence, agents can reason about services to provide automatic service discovery, execution, and composition and inter-operation [McIlraith et al. 2001]. With regard to service discovery, *Semantics* provides matching flexibility and accuracy by considering those concepts that have the same meaning to be similar concepts even though they are syntactically different. In the context of decentralized

[1]http://www.w3.org/Submission/OWL-S/ visited:26/03/2013
[2]http://www.w3.org/2002/ws/sawsdl/ visited:26/03/2013
[3]http://www.w3.org/Submission/WSMO/ visited:26/03/2013

systems for resource location, *Semantics* has been included in several ways: as one of the criteria to organize the network structure, to provide new ways of resource location, and to improve the accuracy of the search results [Haase et al. 2008; Bianchini et al. 2009; Ding et al. 2010; Del Val et al. 2012c; Shaikh et al. 2012; Kontominas et al. 2013].

2.2. Self-Organization

Emergence and organization are close concepts that are used in the context of complex, adaptive systems. We consider *emergence* to be the phenomenon in which global behavior arises from the individual actions of its components. These elementary components do not have a global view of the whole system (i.e., the properties of the system are not present in its components). From our point of view, the concept of *organization* involves an ordered relation among the components. Systems improve their order by making local decisions without a central control that decides how the system evolves. This behavior can be optimized and usually tends to an equilibrium state.

2.2.1. Emergence. *Emergence* is considered to be the phenomenon where global behavior arises from the interactions between the local parts of the system [Wolf and Holvoet 2005; Fernndez et al. 2014]. An example of emergence in decentralized systems such as P2P is the use of *Ant Colony Optimization* algorithms [Caro et al. 2005; Forestiero et al. 2009; Leito 2013]. The inspiration for these algorithms is the behavior of ants and, specifically, a principle called *stigmergy*. Stigmergy is an indirect mechanism of communication that is based on the information that ants leave in the environment. This information is taken into account by other ants in order to make decisions. Works based on stigmergy propose a hybrid protocol for routing and for improving the efficiency of the paths. This protocol combines *reactive ants* (which use broadcast mechanisms for route discovery and bootstrap their routing tables) and *proactive ants* (which use unicast mechanisms based on probabilities for system maintenance).

2.2.2. Organization. *Organization* is considered to be the mechanism that enables a system to arrange its organization at run-time, without explicit external commands. Starting from entities that are structured in a sub-optimal organization or that are not organized at all, an organized system is able to form a specific organization in order to pursue a well-defined goal [Kota et al. 2012]. The main issue in organization is to determine what the best mechanism is to reorganize the current structure through the execution of local actions in order to achieve the desirable behavior when there is a high degree of uncertainty in the system. Therefore, researchers from different areas have proposed mechanisms that deal with the problem of self-organization [Serugendo et al. 2005]. Specifically, we review some of the approaches proposed in Peer-to-Peer and Multi-Agent Systems.

In P2P approaches, there are works where peers consider different criteria to improve the organization of their structural relations. Wang et al. [Wang 2011] present a P2P system where peers use trust values about their neighbors in order to decide which local actions are more appropriate in order to improve the structure of the system. Semantic information and the trust in each peer of the network are taken into account to form groups of peers with similar domains. The system has a hierarchical structure where *expert peers* contain information about the set of peers that have information related to their domain. To build this structure and acquire knowledge about the environment, a broadcast mechanism and trust values are used by the peers. However, the hierarchical organization in peers and *expert peers* can overload the *expert peers* since they initially receive and process all the queries and the system is more vulnerable to deliberate attacks. Condie et. al also consider trust in order to adapt a random network of peers [Condie et al. 2004]. A peer i considers *local trust values*

with respect to each peer that it has interacted with. A *local trust values* represents the number of requests that have been successfully solved by peer j (i.e., the peer that interacts with i). If a peer i has an acquaintance j that has a higher trust value than one of its current neighbors, then it changes its current link for a new one with peer j. In open environments where the peers that are part of the system change it is difficult to establish these successfully because with new peers, peers do not have information about previous direct interactions. There are other approaches that do not use trust mechanisms; instead, they use mechanisms based on tags, gossip, and ostracism to change the structure of the network to avoid relationships with untrustworthy neighbors [Griffiths and Luck 2010; Savarimuthu et al. 2011].

There are still other approaches that instead of considering *trust* they consider *similarity* to decide when the local structure of peers should be reorganized. Raftopoulou and Petrakis present an iCluster overlay network that manages text files. The initial structure of peers is random [Raftopoulou and Petrakis 2008]. The system has two global parameters that establish the number of long-links (links with dissimilar peers) and short-links (links with similar peers) that a peer should have. Periodically, each peer evaluates its degree of internal clustering (degree of similarity of short-links). If the degree of internal clustering is under a certain threshold, the peer initiates a reorganization by sending a message to m of its neighbors in order to find other peers that are similar enough to its interests and to replace its current links. One drawback of this proposal is that nodes initially need to find possible candidates to create clusters through random walks, which affects the success of the searches. Another drawback is that the decision to consider reorganization of the structural links is done periodically instead of when peers consider it to be more appropriate. Also, when peers cannot do a search based on similarity, they use a k-flooding algorithm that increases the traffic in the network. Something else that this approach does not consider is the inclusion of semantic information. Nevertheless, there are other approaches that do consider similarity based on the semantic description of the resources of the nodes of the network. Al-Asfoor et al. propose an initial random structure where nodes self-organize their neighbors by considering a First-In First-Out or semantic criteria [Al-Asfoor et al. 2012]. The authors conclude that a self-organization mechanism based on semantics provides the best results. However, this criterion could divide the system into several isolated clusters that provide similar resources.

Reinforcement Learning has been used in MAS to dynamically adapt the links of agents by calculating a probability that is based on information related to its current state, previous decisions, and environmental conditions [Einhorn and Mitschele-Thiel 2008]. In the self-organization mechanism presented by Abdallah *et al.*, when an agent receives a message, it updates its current state using a reinforcement algorithm and decides whether or not is appropriate to stochastically reorganized its current links by adding or removing neighbors [Abdallah and Lesser. 2007]. The reinforcement learning algorithm that is used in the decision-making process to update the behavior of agents is called the Weighted Policy Learner (WPL). This gradient algorithm allows agents to learn stochastic policies that make agents slow down learning when moving away from a stable policy and speed up learning when moving towards a stable policy. This approach improves previous proposals based on reinforcement learning [Peshkin and Savova 2002] since it considers the dynamism of the network. Nevertheless, the decision-making algorithm considers the reorganization of agent links based on a predefined probability. Moreover, the decision of removing neighbors is also conditioned by a constant that is dependent on the average degree of connection of the network.

There are other approaches that focus on *cooperative problem-solving in organizations* and how these organizations can be rearranged in order to improve their performance as the environmental conditions and the organizational goals change [Kota

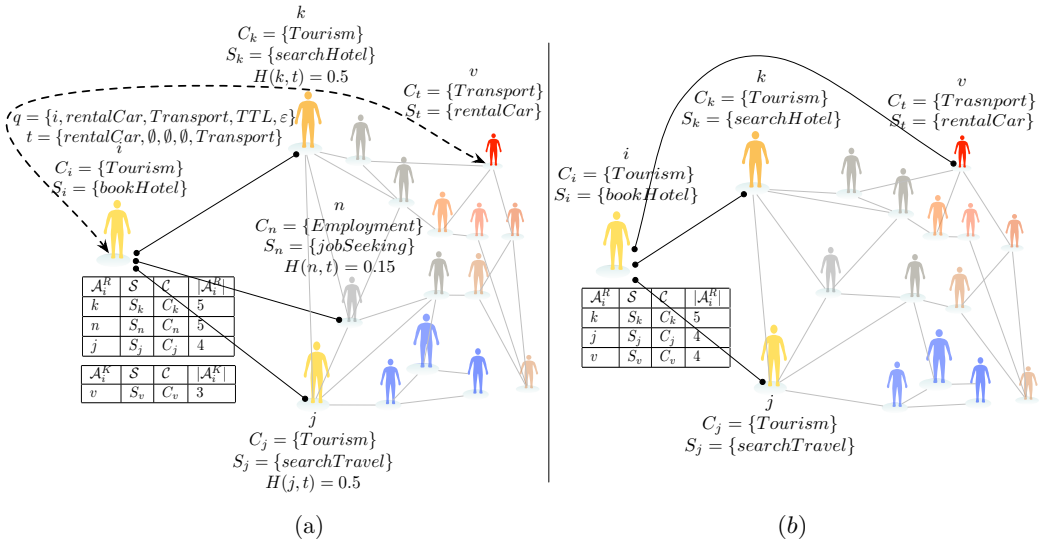
et al. 2012; Kamboj and Decker 2007]. Many of these approaches rely on hierarchical structures where agents change their relations in order to distribute their workload to subordinates. The change of relations is based on a utility function that evaluates the reorganization cost, the load of the agent, and the communication cost. However, some of these models assume that all the agents are acquaintances of each other (a fully connected network), which is not a realistic situation in open environments, and these models also rely on a hierarchical structure that reduces the flexibility of the system. There are other approaches that use self-organization mechanisms to provide a service composition that deals with a specific goal [Khondoker et al. 2011; Nallur and Bahsoon 2012]. The two main drawback of these approaches is that: (i) they assume that there is a global view of all the services available in the system, and (ii) also assume that broadcast mechanisms are used.

In this article, we present a Service-Oriented MAS where the underlying structure is a growing network. When an agent enters the system, it establishes a link with a set of agents that are already present based on a probability that take into account the semantic similarity of the attributes of the agents (i.e., the services that the agents offer). Details can be found in [Del Val et al. 2012a]. The service discovery process carried out by the agents in the system includes a self-organization mechanism to reorganize the structural relations between agents when environmental conditions such as service demand change. Our proposal attempts to improve the mentioned approaches above with regard to structure, service discovery, and self-organization.

From the structural point of view, our system is not based on a hierarchical organization. All the agents are considered to be equal. Our proposal differs from other proposals in the initial self-organization of the network. In the majority of the proposals for decentralized service discovery, the initial structure of the network is random. In our proposal, we present a growing network that is self-organized from the beginning since the connections between new agents and agents that are already present in the system are based on a probability that considers the semantic similarity of the attributes of the agents (i.e., the services of the agents). The semantic service descriptions of the services that the agents offer are public. The semantic similarity between semantic service descriptions is calculated using a matching function that establishes the degree of match between them. The initial network structure that is generated using a self-organization criterion called *homophily* adapts to changes in the service demand through self-organization mechanisms. Also, during the *service discovery process*, local information is taken into account to create the initial structure and to *self-organize* the structural links as service demand changes.

From the service discovery point of view, there is no initial period for acquiring knowledge through flooding strategies. An agent does not maintain information about routes that could change frequently in highly dynamic environments. Each agent only maintains information about who its neighbors are and what services they offer. *Service discovery* is not based on previous information or statistics that require a training period in order to be reliable. It is based on the semantic similarity between the service descriptions of the agents and the degree of connection of the agents. Similarity is calculated by taking into account the semantic information of the agents and not just keywords. We assume that the service descriptions and the degree of connection of an agent is known by its direct neighbors.

From the self-organizing point of view, each agent takes advantage of the information generated during its activity in the service discovery. With this information, agents can reason about when it is more appropriate to consider a structural change in its neighborhood. It is not necessary to have a flooding phase in order to obtain information for the self-organization process. In order to determine which acquaintances might provide a beneficial relation, instead of using global knowledge or randomly se-



$\mathcal{A}_i^R$	$\mathcal{S}$	$\mathcal{C}$	$\lvert\mathcal{A}_i^R\rvert$
k	S_k	C_k	5
n	S_n	C_n	5
j	S_j	C_j	4

$\mathcal{A}_i^K$	$\mathcal{S}$	$\mathcal{C}$	$\lvert\mathcal{A}_i^K\rvert$
v	S_v	C_v	3

$\mathcal{A}_i^R$	$\mathcal{S}$	$\mathcal{C}$	$\lvert\mathcal{A}_i^R\rvert$
k	S_k	C_k	5
j	S_j	C_j	4
v	S_v	C_v	4

Fig. 1. An example of a decentralized service discovery system. (a) Agent i establishes a link with two similar agents k and j and with a dissimilar one n; Agent i only knows its direct neighbors k, j, and n. If agent i needs to locate a service (i.e., rentalCar), it will forward the query to its most promising neighbor (i.e., k) based on the homophily between the neighbor and the target agent (i.e., t) that should provide the required service and the degree of the neighbor. (b) Agent i decides to reorganize its links and changes its link with n (since it is not being used) for a link with a previously known acquaintance v.

lecting neighbor agents use the local view of the traffic of the system. The set of acquaintances that an agent maintains is limited to a fixed number. The agents evaluate the utility of their links by taking into account semantic information about services as well as the information related to the traffic in the network.

3. SERVICE DISCOVERY SCENARIO

To illustrate in what the service discovery and the self-organization mechanism consist of, we present the following scenario. Consider a network of services to be a form of distributed computing system. This network contains different groups of semantic web services that are provided by software agents as a part of an overlay network. The agents offer services that are not appropriate for dealing with their goals. Therefore, they must interact with other agents in order to achieve a task. However, the agents only know their direct neighbors. In order to locate potential provider agents, they need to start an efficient service discovery process that only requires a few steps. Moreover, since we assume that the goals of agents changes over time, the service demand changes and agents consider it to be more beneficial to change their structural relations in order to reach the required provider agents in fewer steps.

The scenario in Figure 1 shows a network of agents that offer semantic web services from different categories. The structural relations between these agents have been established taking a homophily criterion into account. Homophily is a social principle that establishes that contacts between similar people occur at a higher rate than among dissimilar people. Homophily implies distance in terms of social characteristics that can be translated into network distance [McPherson et al. 2001]. In a structure that is based on homophily, an individual has a higher probability of being connected to similar individuals than to dissimilar ones. In the case of agent i, it has connections with agents k and j (which offer similar services) and with agent n (which offers a dis-

similar service). Note that agents that offer services from similar categories are represented in Figure 1 with similar colors. Agent i offers the service *bookHotel*; however, in order to achieve one of its goals, it needs to locate an agent that offers a service similar to $rentalCar$ from the $Transport$ category. At that moment, agent i creates a query q = $\{i, rentalCar, Transport$, TTL, ε $path = \{k\}|k \in \mathcal{A}\}$ that consists of the following: the identifier of the agent that creates the query; the required semantic service description; the semantic category associated to the service; the Time To Live (TTL), which is the maximum number of times that the query can be forwarded; the threshold that indicates the degree of similarity that an agent takes into account to stop the search; and the set of agents that participate in the search process to reach the target agent. If the query exceeds the TTL, it is considered to be a failure of the service discovery process. Otherwise, the query is forwarded to one of the neighbors. It is assumed that all the agents are collaborative and follow the same criterion to forward the queries.

In the scenario shown in Figure 1a, agent i should choose one of its neighbors,n, j, or k, to forward the query q. In order to select the most promising neighbor, the agent i considers: (i) the homophily between its neighbors and a *fictitious* agent $t = (rentalCar, Transport)$ that offers a service similar to the service that appears in the query and that has a category that is similar to the category specified in the query q; and (ii) the degree of connection of the neighbors. Assuming the values of homophily that appear in Figure 1a and the degree of connection of each neighbor, agent i sends the query to the most promising agent (i.e., agent k). This process is repeated until the semantic similarity between a local service of an agent and the service in the query is over a certain threshold ε or the query exceeds the TTL. In the described scenario, the process ends when the query arrives to agent v (see Figure 1a). Afterwards, agent i stores agent v in its local view as a possible candidate for establishing a future structural relation if some of its current relations are not being used. Since the number of acquaintances is limited, an agent maintains an acquaintance in the acquaintance set until it is completed. If the set is completed and a new acquaintance is considered to be added to the set, an existing acquaintance is replaced by the new one based on its utility estimation.

As time passes, service demand changes. Based on its local view of the system, agent i realizes that the service demand has changed and that the link with agent n is not being used to forward queries. However, it has an acquaintance v that connects to a set of agents that offer services that are being demanded at that moment. Therefore, agent i decides to break its current structural relation with n and establishes a new one with an acquaintance that was discovered as a result of a previous service discovery process ($(i, n) \rightarrow (i, v)$) (see Figure 1b). This self-organization action reduces the path distance towards agents that provide demanded services and also improves the success rate in future discovery processes.

4. A FORMAL MODEL FOR SERVICE-ORIENTED MAS

In order to deal with the decentralized service discovery and self-organization process described in the above scenario, we propose a decentralized model that is made up of a set of autonomous agents that offer their functionality through a set of semantic services. These agents have a reduced view of the global community: just a limited number of direct neighbors are known and the rest of the network remains invisible to them. We assume that each relation with a neighbor implies a maintenance cost; therefore, agents have a limited number of relations. By simply considering local information, agents are able to locate the required service and update their structural relations with other agents in order to adapt to changes in the service demand.

DEFINITION 4.1. *(System). The system is a tuple* $< \mathcal{A}, L >$*, where* $\mathcal{A} = \{i, ..., n\}$ *is a finite set of autonomous agents and,* $L \subseteq \mathcal{A} \times \mathcal{A}$ *is a set of links, where each link* $(i, j) \in L$ *indicates the existence of a direct relationship between agent* i *and* j*.*

It is assumed that the knowledge relationship among agents is symmetric; therefore, the network is an undirected graph. An agent controls its own information about (i) the semantic services that it offers, (ii) the categories of its services, and (iii) local knowledge about a set of neighbors and acquaintances.

DEFINITION 4.2. *(Agent). The knowledge model of agent* i *is a tuple* $< \mathcal{S}_i, \mathcal{C}, \mathcal{A}_i^R, \mathcal{A}_i^K, \phi_i >$*, where*

- *—*$\mathcal{C}$ *is the set of service categories (relation types).*
- *—*$\mathcal{S}_i = \{s_1, \ldots, s_k\}$ *is the set of semantic services offered by the agent* i*. Each service has an associated category* $c \in \mathcal{C}$*. Each service* $s_k \in \mathcal{S}_i$ *is defined by the tuple* $s_k = (I_k, O_k, P_k, Eff_k)$*, where* I_k *is the set of inputs,* O_k *is the set of outputs,* P_k *are the preconditions for the execution of the service, and* Eff_k *are the effects of the service execution.*
- *—*$\mathcal{A}_i^R \subseteq \mathcal{A}$ *is the set of agents that agent* i *has a structural relation with. Each agent* i *maintains a vector of values* $\boldsymbol{\tau}_{i,j} = [\tau_{i,j}^1 \ ... \ \tau_{i,j}^{|\mathcal{C}|}]$ *for each one of its neighbors. From the point of view of agent* i*, an element* $\tau_{i,j}^c$ *of the vector represents the utility of the relation between* i *and* j *for queries of category* c*. Also, agent* i *knows the set of services that each neighbor offers.*
- *—*$\mathcal{A}_i^K \subseteq \mathcal{A}$ *is the set of acquaintances of agent* i*. If agent* j *is an acquaintance of agent* i*, it means that* i *is at least aware of the existence of* j*. In the context of our service discovery scenario, acquaintances are the agents that are found as a result of the discovery process but agent* i *does not have a direct link with them. Agent* i *maintains a vector of values* $\boldsymbol{\eta}_{i,j} = [\eta_{i,j}^1 \ ... \ \eta_{i,j}^{|\mathcal{C}|}]$ *which represents the probability that agent* i *will establish a new relation of category* $c \in \mathcal{C}$ *with agent* j*.*
- *—*ϕ_i*:* $\mathcal{A} \rightarrow \mathcal{A}_i^R$ *is the forwarding function that selects the most promising neighbor to forward a service request to during the service discovery process.*

The main focus of our self-organization mechanism is the adaptation of structural relations. Structural relations define the set of agents with which an agent establishes a relation. The criterion considered to initially establish structural relations is *homophily* [Lazarsfeld 1954; McPherson et al. 2001]. This criterion is present in many Complex Networks and has been used in the system presented in this paper to create the social structure of agents in a self-organized way. Homophily allows agents to establish structural relations when they enter the system and do not have previous information with which to estimate the utility of potential structural relations. The effects of the homophily criterion to establish links is a network where agents are usually connected with similar agents and also with a few dissimilar agents. This structure facilitates the decentralized search of services by reducing the number of steps needed to locate a resource [Del Val et al. 2012c].

In our system, the *homophily* function $H(i, j)$ calculates the similarity of two agents i and j based on the degree of match between two sets of services, where $\mathcal{S}_i$ and $\mathcal{S}_j$ are the sets of services provided by the agents i and j, respectively. We consider each set of services $\mathcal{S}_i$ (or $\mathcal{S}_j$) to be composed of a set of semantic concepts that can be classified as: Inputs (I_i), Outputs (O_i), Preconditions (P_i), and Effects (Eff_i).

The level of matching between two sets of semantic concepts, C_i and C_j, is calculated through a *bipartite matching graph* [Bellur and Kulkarni 2007] (see Figure 2). Let $G = (C_i, C_j, E)$ be a complete, weighted bipartite graph that links each concept $c_i \in C_i$ to each concept $c_j \in C_j$, $(c_i, c_j) \in E$, and let E represent the edges established in the graph

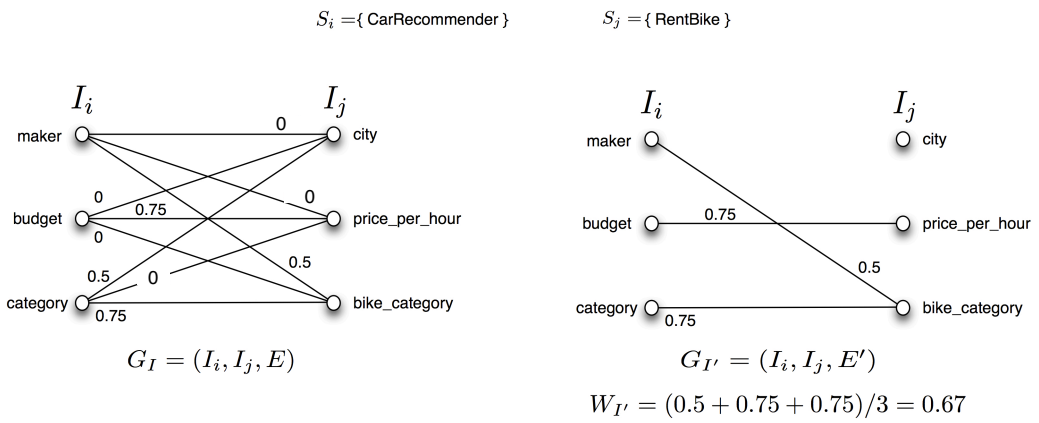


Fig. 2. (Left) Full connected weighted bipartite graph G_I, and (Right) resulting maximum weighted matching relaxed bipartite graph G'_I.

$E = C_i \times C_j$. The term ω_{ij} represents the weight associated to the arc $e_i = (c_i, c_j) \in E$ between c_i and c_j as the semantic similarity between those concepts. Four degrees of match can be identified: *exact, subsumes, plug-in*, and *fail* [Paolucci et al. 2002]. The match is considered to be *exact* if $c_1 \in C_i$ is equivalent to $c_2 \in C_j$ ($c_1 \equiv c_2$); it is *subsumes* if c_1 subsumes c_2 ($c_1 \sqsupset c_2$); it is *plug-in* if c_1 is subsumed by c_2 ($c_1 \sqsubset c_2$); and it is *fail*, otherwise. For simplicity, we have considered these four degrees of match, but other degrees could be considered [Klusch et al. 2009]. A value in the interval $[0, 1]$ is assigned to each degree of match, where 1 represents an exact match between the terms, 0.75 represents a subsumes relation, 0.5 represents a plug-in relation, and 0 represents a fail. The best match among concepts is obtained by calculating the *maximum weighted bipartite matching*, $G' = (C_i, C_j, E')$, where $E' \subseteq E$ are the edges that have the maximal value. The graph G' is a relaxed bipartite graph because not all the concepts from C_j have to be connected to a concept in C_i; therefore, two concepts from C_i can share a concept from C_j. The weight of this graph is calculated as follows:

$$W_{G'} = \frac{\sum_{\omega_{ij} \in E'} \omega_{ij}}{\max(|C_i|, |C_j|)} \tag{1}$$

Specifically, to calculate the *homophily* between two agents, four bipartite graphs are defined, (one for each of the components of services present in the sets $\mathcal{S}_i$ and $\mathcal{S}_j$): Inputs (I_i, I_j), Outputs (O_i, O_j), Preconditions (P_i, P_j), and Effects (Eff_i, Eff_j). The linear combination of the $W_{G'}$ of each set of concepts gives the value of the homophily between agents (see Equation 2, where the parameters α and β assign different weights to the components of the formula).

$$H(i,j) = \alpha\left[\beta * W_{G'_I} + (1-\beta)W_{G'_O}\right] + (1-\alpha)\left[\beta * W_{G'_P} \qquad +(1-\beta)W_{G'_{Eff}}\right] \tag{2}$$

The homophily between agents is used to build a network that is based on preferences, which grows according to a simple self-organized process. The construction process of a growing network ensures that the oldest nodes have a higher probability of receiving new links than the newest ones. Therefore, the total number of neighbors that an agent has will depend on agent's age. The average degree of connection of a

network that is built following this process follows an exponential distribution [Dorogovtsev and Mendes 2003].

5. SELF-ORGANIZATION IN SERVICE DISCOVERY

In the proposed decentralized system, when an agent needs a service, since there is no service discovery facilitator nor a registry to be queried, an active search process must be launched. This search process determines whether or not there is an agent in the network that provides a service that is similar enough to the one required. Depending on the success of the query resolution, an agent reasons about whether it is worthwhile to maintain or change its current structural relations. In this section, the service discovery and the self-organization of the structural relations are explained.

5.1. Service Discovery

The process of service discovery is carried out as follows. An agent i sends a query q that contains the identifier of the agent, the semantic service description, the category of the required service, the Time To Live (TTL) (which is the maximum number of times that the query can be forwarded), the similarity threshold to consider a service that is similar enough to the target service, and the agents that participate during the service discovery ($q = \{i, s, c, TTL, \varepsilon\}$,$path = \emptyset$). Then, the query is forwarded to the most promising agent among its neighbors $j \in \mathcal{A}_i^R$ (i.e., the agents with which agent i has a structural relation).

$$\phi_i(t) = \underset{j \in \mathcal{A}_i^R}{\operatorname{argmax}} \left[1 - \left(1 - \left(\frac{H(j,t)}{\sum_{n \in \mathcal{A}_i^R} H(n,t)} \right) \right)^{|\mathcal{A}_j^R|} \right] \tag{3}$$

Equation 3 calculates the most promising neighbor $j \in \mathcal{A}_i^R$ of an agent i in order to reach an initially unknown provider agent t that has the service s of the query $q = \{id, s, c, TTL, \varepsilon, path = k \mid k \in \mathcal{A}\}$ in its set of services $\mathcal{S}_t$. This equation uses homophily-based factors (H) and degree-based factors (number of neighbors $|\mathcal{A}_j^R|$) to explore the network. The divisor of the expression is just a normalization factor. The homophily-based factor is based on the semantic similarity between the services offered by the agent j and the service that the target agent t should offer (i.e. the service s specified in the query q). As an example, in Figure 1 agent i has three neighbors to forward the query to (k, j, n). In order to select the most promising neighbor, agent i applies Equation 3 as follows:

$$\phi_i(t) = \underset{k,j,n}{\operatorname{argmax}} \left[1 - \left(1 - \frac{0.5}{1.15} \right)^5, 1 - \left(1 - \frac{0.5}{1.15} \right)^4, 1 - \left(1 - \frac{0.15}{1.15} \right)^5 \right] = \underset{k,j,n}{\operatorname{argmax}} [0.942, 0.897, 0.502] = k$$

This decision minimizes the length of the path to the provider agent that can solve the query since the structure of the network is based on degree and homophily [Del Val et al. 2012c].

The receiver agent updates its information about the queries received (see Alg. 1 Lines 2,3). Then, if the TTL of the query does not exceed the TTL, the receiver agent performs a matchmaking of the query against the services it offers. If the best matching service has a degree above a certain semantic similarity threshold ε, then the

search ends successfully (see Alg. 1 Line 6). This threshold is established by the agent that starts the service discovery.

In the case that the target agent is found, the agent that started the process adds the provider agent as an acquaintance agent (see Figure 1b). The provider agent that has the required service propagates a message to the agents that participated in the search, which in turn update and analyze the utility of their relations (see Alg. 1 Lines 9,10).

Following the same process, in the case of an unsuccessful matching, the agent decreases the TTL and forwards the query to its most promising neighbor. Each time an agent forwards a query, it adds its identification to the query (see Alg. 1 Lines 12,14).

ALGORITHM 1: Function that describes the service discovery process that an agent i carries out when it receives a query.

1: **function** $serviceDiscovery(i, q = (id, s, c, TTL, \varepsilon, path))$
2: $\#_i \leftarrow \#_i + 1$ /* num queries received */
3: $\#_i^c \leftarrow \#_i^c + 1$ /* num queries of category c that agent i received */
4: $t \leftarrow (s, c, \emptyset, \emptyset)$ /* target agent that provides service s of category c */
5: **if** $TTL > 0$ **then**
6: /* checks the homophily of agent i with the target agent t */
7: **if** $H(i, t) \geq \varepsilon$ **then**
8: /* the agent i offers a service similar enough to the service of the target agent t */
9: /* agent i sends an inform message to the agent id that initiates the search*/
10: $inform(id, i)$
11: $\mathcal{A}_{id}^K \rightarrow \mathcal{A}_{id}^K \cup \{i\}$
12: $updateLinksUtility(path)$ /* agents that participate in a successful search update their information*/
13: $selfOrganization(path)$
14: **else**
15: $i \leftarrow \phi_i(t)$ /* selects the most promising neighbor */
16: $TTL \leftarrow TTL - 1$
17: /* the query is forwarded to the most promising neighbor */
18: $serviceDiscovery(i, q = (id, s, c, TTL, \varepsilon, path \cup \{i\}))$
19: **end if**
20: **else**
21: $inform(id, \emptyset)$
22: **end if**
23: **end function**

5.2. Self-Organizing Structural Relations

The structure of relations, that is, how agents are arranged, may severely affect the performance of the system [Abdallah and Lesser. 2007; Gaston and desJardins 2005; Kota et al. 2012]. Therefore, agents should check their structural relations with a frequency that is based on the number of changes in the environmental conditions. Each agent considers two aspects: (i) the establishment of suitable criteria to evaluate its structural relations; and (ii) when it is more appropriate to change its relations by breaking one or more relations or by establishing new relations with its acquaintances.

The criterion that we propose to evaluate, *structural relations*, is based on their *utility*. In the context of service discovery, we have defined the utility of a structural relation between agents i and j for a category c as:

$$U_{i,j}^c = \frac{\#_i^c}{\#_i} \cdot m_j^c, \tag{4}$$

where $\frac{\#_i^c}{\#_i}$ is the ratio between the number of queries for service category c that were received by i and the total number of queries received by i so far, and $m_j^c \in (0, 1)$ is

the average degree of match for queries of category c performed by agent j so far. Note that $\frac{\#_i^c}{\#_i}$ represents how important c is for agent i, while m_j^c reflects the specialization of j in the services that belong to category c. Note that the term m_j^c and the term H of Equation 3 (which is used to select the most promising neighbor) are related since H also considers the specialization of its neighbors in services that are similar to the services in the query.

As time passes, agent i evaluates probabilistically whether to maintain a relation with agent j for queries of category c. If a relation with a neighbor is frequently and successfully used to redirect queries about services of a certain category, then it is interesting for the agent to maintain the relation. However, if a relation is seldom used, then the agent must decide whether or not to maintain it. Therefore, relations that are used during the discovery process are continuously reinforced by productive interactions, while other relations are weakened and eventually broken. The utility of a structural relation decays exponentially according to Eq. 5 [Jin et al. 2001]:

$$\tau_{i,j}^c = 1 - e^{-\rho \cdot U_{i,j}^c} \tag{5}$$

where $\rho \in (0, \infty)$ is an adjustable parameter and $U_{i,j}^c \in \mathbb{R}^+$ is the utility of the established relation between agent i and agent j for the category c. High values of ρ make $\tau_{i,j}^c$ approximate 1 even with low values of utility, while low values of ρ make the agent more demanding and the relation must have a high utility in order to be maintained (see Fig. 3).

Each agent i maintains a vector of values $\boldsymbol{\tau}_{i,j} = [\tau_{i,j}^1 \ ... \ \tau_{i,j}^{|\mathcal{C}|}]$ for each one of its neighbors. From the point of view of agent i, an element $\tau_{i,j}^c$ of the vector represents the utility of the relation between i and j for queries of category c. When agent i establishes a new relation of category $c \in \mathcal{C}$ with agent j, the corresponding value $\tau_{i,j}^c$ is initialized to 1.

Since established relations may break over time, new relations with some of the acquaintances can be formed. Thus, for every acquaintance $j \in \mathcal{A}_i^K$, agent i maintains a vector of values $\boldsymbol{\eta}_{i,j} = [\eta_{i,j}^1 \ ... \ \eta_{i,j}^{|\mathcal{C}|}]$ that represents the probability that agent i will establish a new relation of category $c \in \mathcal{C}$ with agent j. The value $\eta_{i,j}$ is strengthened every time agent i obtains some knowledge about agent j that increases the potential utility of establishing a relation with agent j, and it is weakened every time agent i becomes aware of something about the acquaintance j that decreases this potential utility.

Let $U_{i,j}^c$ be the estimated utility that agent i would obtain if it established a relation of type c with agent j. The probability of actually establishing a new relation with agent j is given by Eq. 6 (note the similarity to Eq. 5):

$$\eta_{i,j}^c = 1 - e^{-\mu \cdot U_{i,j}^c} \tag{6}$$

where $\mu \in (0, \infty)$ is another adjustable parameter. The greater the estimated utility by including a relation of category c with agent j, the higher the probability that this relation will actually be established.

Agents that participate in the service discovery update their information about the utility of their relations with other agents taking into account the information about their current links with neighbors and acquaintances (see Alg. 2). Afterwards, the agents reason about changing their structural relations. This reasoning process is described in Algorithm 3. An agent i that has participated in a successful search process

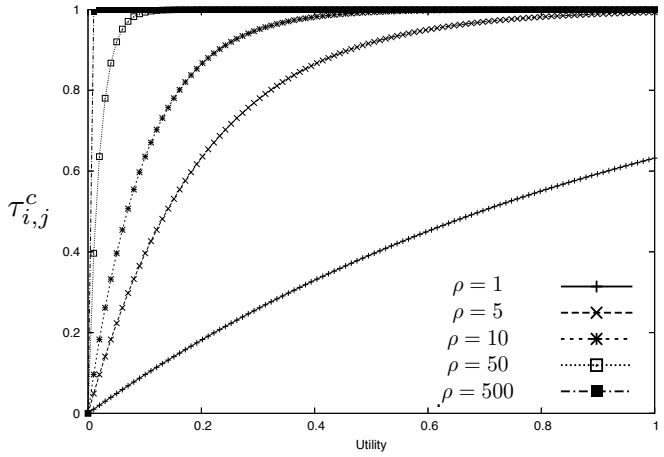


Fig. 3. Functional form of $\tau^c_{i,j}$ for different values of ρ. When $\rho = 1$, agents decide to change their links as soon as their utility decreases. As the value of ρ parameter increases, agents maintain links with low utility until their utility is close to 0. At that moment, their utility and the probability of maintaining the links decreases faster.

ALGORITHM 2: Function that describes how an agent i updates the utility of its links.

function $updateLinksUtility(path)$
for $i \in path$ **do**
  /* agents that participate in a successful search process */
  for $j \in \mathcal{A}_i^K \cup \mathcal{A}_i^R$ **do**
    for $c \in \mathcal{C}$ **do**
      $U^c_{i,j} \leftarrow \frac{\#^c_i}{\#_i} \cdot m^c_j$
      if $j \in \mathcal{A}_i^R$ **then**
        $\tau^c_{i,j} \leftarrow 1 - e^{-\rho \cdot U^c_{i,j}}$ /* neighbors */
      else
        $\eta^c_{i,j} \leftarrow 1 - e^{-\mu \cdot U^c_{i,j}}$ /* acquaintances */
      end if
    end for
  end for
end for
end function

determines the category c based on the maximum number of queries that it received. Then, the agent selects the neighbor $j \in \mathcal{A}_i^R$ with the minimum utility value for this category ($minU(\mathcal{A}_i^R)$) and the acquaintance $ai \in \mathcal{A}_i^K$ with the maximum utility value for this category ($maxU(\mathcal{A}_i^K)$). After that, in order to consider a structural change, agent i checks if the utility of the acquaintance ai is greater than the utility of its current neighbor j. Agent i also checks if the current neighbor j has a degree of connection of 3.

The structural relations between agents are undirected. Therefore, in our system, both agents that are in a relation must agree that a change in their relation is appropriate. Agent j analyzes which is the category c with the maximum number of queries that it received.Then, the agent selects the neighbor $n \in \mathcal{A}_j^R$ with the minimum utility

value for this category ($minU(\mathcal{A}_j^R)$) and the acquaintance $aj \in \mathcal{A}_j^K$ with the maximum utility value for this category ($maxU(\mathcal{A}_j^K)$). Similarly to agent i, agent j checks if the utility of the acquaintance aj is greater than the utility of its current neighbor n. Agent j also checks if its current neighbor i has a degree of connection of 3.

In order to change the link (i, j) for a new one, the algorithm determines which agent will obtain the highest utility when a new link is established. That agent will be the one that changes its current structural relation with the previously selected acquaintance.

ALGORITHM 3: Function that describes how an agent i decides to reorganize its links.

```
1: function selfOrganization(path)
2: for i ∈ path do
3:     c ← argmax(#_i^c | c ∈ C)
4:     j, minU_i ← minU(A_i^R)          /* i's neighbor with the lowest value of τ_{i,j}^c */
5:     ai, maxU_ai ← maxU(A_i^K)        /* i's acquaintance with the highest value of η_{i,ai}^c */
6:     if (maxU_ai > minU_i) ∧ (|N_j| > 2) then
7:         c ← argmax(#_j^c | c ∈ C)
8:         n, minU_j ← minU(A_j^R)      /* j's neighbor with the lowest value of τ_{j,n}^c */
9:         aj, maxU_aj ← maxU(A_j^K)    /* j's acquaintance with the highest value of η_{j,aj}^c */
10:        if (maxU_aj > minU_j) ∧ (|N_i| > 2) then
11:            if maxU_ai > maxU_aj then
12:                A_i^R ← A_i^R − j
13:                A_i^K ← A_i^K − ai
14:                A_i^R ← A_i^R ∪ {ai}
15:            else
16:                A_j^R ← A_j^R − n
17:                A_j^K ← A_j^K − aj
18:                A_j^R ← A_j^R ∪ {aj}
19:            end if
20:        end if
21:    end if
22: end for
23: end function
```

6. EXPERIMENTS

In order to evaluate the proposed mechanism for self-organization during service discovery in Service-Oriented MAS, we performed several tests. We developed our own simulation tool in Java to validate our proposal. In the experiments, we did not focus on how much time each simulation required since we considered that the number of snapshots that each self-organization mechanism requires would be less dependent on the hardware where the experiments were performed. The number of iterations that we considered in each experiment was established based on the evolution of the system until the results remained constant.

The tests that we used to evaluate the performance of the service discovery when self-organization mechanisms are used by the agents. The tests include the following metrics:

— the average number of steps that are needed to successfully resolve a query,
— the percentage of successfully resolved queries,
— the number of structural relations that have changed during the service discovery,
— the efficiency of the system, calculated as

$$E = \frac{\#q \cdot p}{\#msg} \cdot \frac{\#l}{\#l + \Delta l} \tag{7}$$

The rationale of Equation 7 is the following. Let $\#q$ be the number of queries that have been successfully solved, p the average number of steps required to arrive to the target service, $\#msg$ the total number of messages generated, $\#l$ the number of the original relations, and Δl the number of structural changes that have occurred as a result of the adaptation decisions in the system. The first term of Equation 7 indicates the degree of adaptation of the system. When the value of this term approaches 1, it means that the system is completely adapted. In this case, the majority of searches end successfully, and the number of messages generated in the system is close to the number of messages in the best scenario ($\#q \cdot p$). If the system is not adapted to the service demand, the number of unsuccessful searches increases, and, therefore, the number of useless messages that overload the system increases. Consequently, this term, which reflects the adaptation of the system, is close to 0. The second term of the efficiency metric indicates the quality of the structural changes. The combination of the two terms reflects the efficiency of the system. The system efficiency is high (close to 1) when the number of structural changes is low (according to a certain threshold) but high enough to do the following: reduce the path length, improve the number of successful searches, and reduce the number of useless messages in the system. The system efficiency is low when there is a high number of structural changes but the path length is not reduced. Therefore, the number of useless messages that navigate the network increases the workload of the system and reduces its efficiency.

Each network of the tests that we performed was an undirected network based on homophily with 1,000 agents. We considered 30 networks in each test. We assumed that all the agents were cooperative and had a homogeneous behavior, that is, the agents would fulfill the rules and redirect the query. Each agent offered one semantic web service associated to a category. The agents were distributed over 16 semantic categories. The set of semantic service descriptions used for the experiments were taken from the test collection OWL-S TC4.[4]

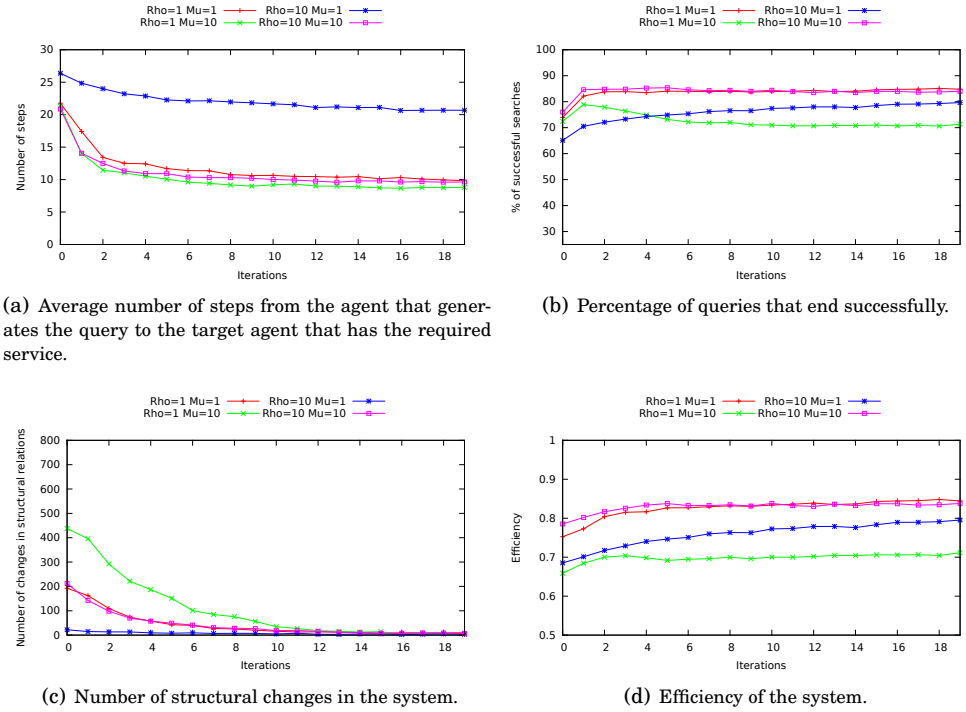
All the agents in the system had the same probability of generating service queries. The query was successfully solved when an agent that offered a similar service (over a threshold ε) was found before the TTL expired ($TTL = 100$).

6.1. Self-Organization Parameters

In this test, we analyzed the influence of the ρ and μ parameters (see Eqs. 5 and 6) in the self-organization mechanism proposed in this paper. The average degree of connection of the network was 2.5. Query distribution in this test was modeled as an exponential distribution ($\lambda = 0.35$) that represents that there are always a few service categories that are the most demanded and the rest of services have a lower demand rate [Adamic and Huberman 2002; Huberman and Adamic 2000]. In the experiment, we made a snapshot of the measures in each iteration. Each snapshot consisted of 5,000 queries. The value of ε threshold was 0.75.

We considered several combinations of ρ and μ values that represent different self-organization behaviors of agents. We grouped these combinations into 4 cases. Each case defines an adaptation behavior of the agents. These behaviors range from 'impulsive' behaviors where agents rewire their links as soon as their utility decreases to more 'rational' or 'demanding' agents that wait until the utility of their links decreases

[4]http://www.semwebcentral.org/projects/owls-tc/ visited:15/10/2012



(a) Average number of steps from the agent that generates the query to the target agent that has the required service.

(b) Percentage of queries that end successfully.

(c) Number of structural changes in the system.

(d) Efficiency of the system.

Fig. 4. Evolution over time of system measures considering different values for the ρ and μ parameters.

ρ (neigh.) / μ (acq.)	not demanding	strict
not demanding	A ($\rho = 1, \mu = 10$)	B ($\rho = 10, \mu = 10$)
strict	C ($\rho = 1, \mu = 1$)	D ($\rho = 10, \mu = 1$)

Fig. 5. Influence of ρ and μ parameters when these parameters take values of 1 or 10 in the structural self-organization.

considerably. The results are shown in Figure 4. We show the most representative values of ρ and μ for each case (see Table 5 and Fig. 3):

— Case A ($\rho = 1, \mu = 10$). The expression $\rho = 1$ means that the agent decides to rewire a relation quickly as soon as its utility starts to decrease; $\mu = 10$ implies that the agent is not strict with the utility of the acquaintances. An agent with that configuration is not rigorous with the utility of its current relations, and, as soon as their utility decreases, it will replace them with acquaintances that it is not sure that will be used for the forwarding process. Therefore, the structural changes are almost random. In this scenario, the number of steps in the discovery process decreases since only the queries about services that are situated near the source agent are solved. The improvement in the success rate and efficiency is not significant due to the random and high number of structural changes that do not provide a suitable reorganization of the structural relations.

— Case B ($\rho = 10, \mu = 10$). The expression $\rho = 10$ means that the agent decides to maintain its current relations even though their utility has low values; $\mu = 10$ implies that

the agent is not strict with the utility of the acquaintances. An agent with this configuration rewires its current relations when its utility has decreased considerably. The agent replaces them with acquaintances without high utility. In this scenario, the number of rewired relations is low, the average number of steps in the discovery process decreases considerably in the first iterations, and there is an improvement in the success rate and efficiency.

— Case C ($\rho = 1, \mu = 1$). The expression $\rho = 1$ means that the agent decides to remove a relation as soon as its utility starts to decrease; $\mu = 1$ means that the agent is strict with the utility of its acquaintances. The agent does not consider an acquaintance to be a *good alternative* if it does not have a utility value that is high enough. Even though the agent wants to rewire its relations when their utility starts to decrease, it has to wait for an acquaintance with high utility. In this scenario, the number of rewired relations is a bit lower than expected, but there is an improvement in the success rate and efficiency.

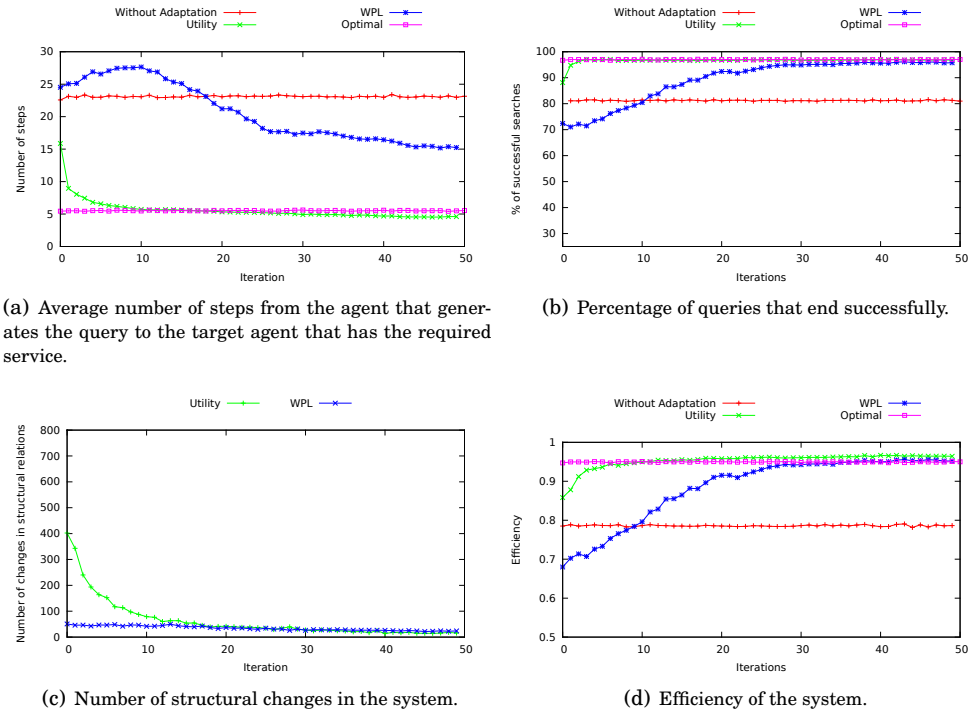
— Case D ($\rho = 10, \mu = 1$). The expression $\rho = 10$ means that the agent decides to maintain its current relations until their utility has low values; $\mu = 1$ means that the agent is strict with the utility of the acquaintances. In this scenario, the agent only rewires relations when their utility is really low and if their acquaintances have a high utility. This configuration produces the lowest number of structural changes. Therefore, the improvement in the service discovery process and in system efficiency takes more time.

From the results shown in Figure 4, we can conclude that, in configuration A, the agents rewire many more relations than necessary to adapt the system to the service demand. Each structural change implies a cost for the system; therefore, its efficiency decreases considerably (see Eq. 7). In configuration D, the agents are not impulsive and they decide to wait until the utility of their links decreases. Therefore, the adaptation process does not consider many structural changes. The degree of adaptation achieved is not enough to provide a significant improvement in the different set of measures (path length, percentage of successful searches, and efficiency). This configuration is not appropriate in dynamic environments where the service demand changes frequently. In scenarios B and C, there is a balance between the number of rewired relations and the improvement in system performance. It can be concluded that the best configurations are B and C.

6.2. Comparison with other approaches

In the second test, we evaluated the influence of our self-organization mechanism based on: (i) the utility functions for the evaluation of structural relations with neighbors, $\mathcal{A}_i^R$, and acquaintances, $\mathcal{A}_i^K$ (see Eqs. 5 and 6); and (ii) the criteria that each agent uses to decide when it is more appropriate to change current structural relations. Figure 6 shows the comparison of the following: the results of our proposal (`Utility`), the results without using adaptation mechanisms (`Without Adaptation`), the results obtained with a system where the service distribution over agents is equal to the service demand (`Optimal`), and the results with a self-organizing mechanism based on reinforcement learning algorithm (`WPL`) (see Figure 6).

The self-organizing mechanism based on RL that is shown in this comparison uses a learning strategy (Weighted Policy Learner) that is similar to WoLF [Bowling and Veloso 2002]. The Weighted Policy Learner algorithm is based on the following idea: slow down learning when moving away from a stable policy and speed up learning when moving towards a stable policy. The decision-making algorithm for establishing when it is appropriate to add or remove a link is based on a reorganization parameter and on the average degree of connection of the network.



(a) Average number of steps from the agent that generates the query to the target agent that has the required service.

(b) Percentage of queries that end successfully.

(c) Number of structural changes in the system.

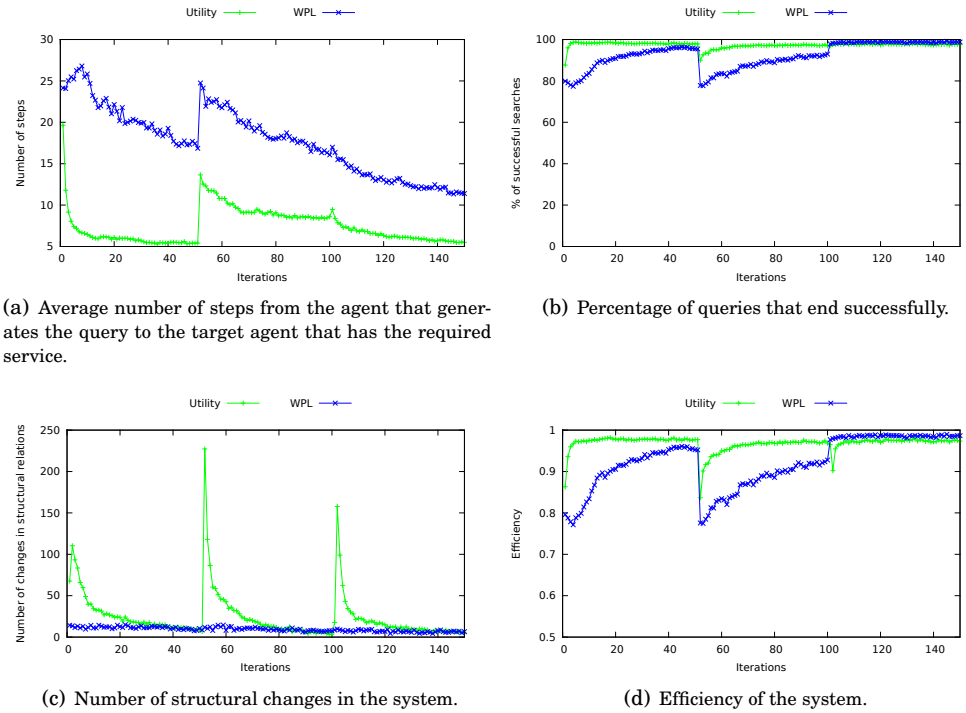
(d) Efficiency of the system.

Fig. 6. Evolution over time of system measures considering different adaptation mechanisms: `Utility` and `WPL`.

The experiments were done in 30 networks where the average degree of connection of the networks was 4, since `WPL` broke the networks into too many isolated parts when the average degree of connection was 2.5. In this test, the number of queries per iteration was 5,000. The distribution of queries followed an exponential distribution ($\lambda = 0.35$). `WPL` had a reorganization rate value of 0.002. `Utility` was configured with parameters $\rho = 1$ and $\mu = 1$. Note, that for reasons of clarity, the error intervals of the results of the experiments are not shown here in the graphs.

In general, both strategies improved the average path length in the search process (see Fig. 6(a)). However, `WPL` took more time to reduce the average path length in the searches, and its improvement was not as significant as the improvement achieved by `Utility`. Note that in the Figure 7 the error intervals are bigger with `WPL` since the degree of adaptation to the service demand achieved is lower than the degree of adaptation achieved with `Utility`. The error intervals of the `Utility` strategy decreases as the degree of adaptation increases. At the 10th iteration, the mean path and the error intervals are equal to those obtained with the `Optimal` system adaptation.

Considering the number of changes in the structural relations between agents (see Fig. 6(c)), `Utility` initially generates a high number of changes if we compare it with `WPL`. In fact, `WPL` follows a constant rate of changes, and the adaptation is slower. With `Utility` the agents only rewire relations when the acquaintances are significantly better than the current relations. This makes agents change a reasonable number of structural relations. Through local decisions of agents, the system is able to regulate the number of structural changes required. As the structure is getting adapted to the service demand, the number of changes decreases and so does its variability (i.e., the error intervals are smaller than in the first interactions). The success of the service



(a) Average number of steps from the agent that generates the query to the target agent that has the required service.

(b) Percentage of queries that end successfully.

(c) Number of structural changes in the system.

(d) Efficiency of the system.

Fig. 7. Evolution over time of system measures considering dynamic changes in the service demand and different adaptation mechanisms: `Utility` and `WPL`.

discovery system is improved with both strategies (see Fig. 6(b)). With both adaptation mechanisms, agents were able to create new relations that connect them with other agents that offer the most demanded services. `Utility` improved the success rate in the first two iterations. However, `WPL` took more time to achieve a success rate over 90%. Fig. 6(d) shows the efficiency of the system when self-organization mechanisms are included. The efficiency was calculated taking into account the success of the service discovery, the average path length, and the number of structural changes (see Eq. 7). The best results were obtained by the `Utility`, which reduces the number of steps in the search process and increases the number of successful discovery queries faster than `WPL`. Moreover, `Utility` is able to determine in a decentralized way whether or not it is appropriate to make structural changes.

6.3. Dynamic Service Demand

Since agent activity evolves over time (i.e., according to the time of day, the different days of the week, or the different seasons of the year [Howard et al. 2001][Aquin et al. 2010]), the system should be able to adapt itself without external coordination based on to the customers' demand dynamics. The aim of the third test was the evaluation of the performance of adaptation mechanisms with dynamic service demands.

In this test, the number of queries per iteration was 1,000. The experiments were done in 30 networks where the average degree of connection of the networks was 4, since `WPL` breaks the networks into too many isolated parts when the average degree of connection was 2.5. `WPL` had a reorganization rate value of 0.002. `Utility` was configured with parameters $\rho = 1$ and $\mu = 1$ (see Section 6.1 case B).

Initially, the service demand followed an exponential distribution where there was a reduced set of service categories that were much more demanded than the other categories. This demand changed at iteration 50. The new demand followed another exponential distribution, but after iteration 50 the most demanded services were from categories that were the least demanded in the previous iterations. This new distribution continued until iteration 150, where the service demand was reverted to the initial distribution.

Figure 7 shows the results of this experiment. In the first interval [0,50], `Utility` allows agents to adapt their structural relations faster; therefore, the number of steps in the discovery process is reduced, the success improves, and the system efficiency increases considerably even though the number of structural changes is high. `WPL` needs more time to adapt the structural relations to the current demand since the number of redirections is too low to deal with changes in the service demand. Note that since the degree of adaptation achieved with `WPL` is not as high as the degree of adaptation achieved by `Utility`, the error intervals with `WPL` are bigger than with `Utility`. As this Figure 7 shows, at the beginning of the second interval [50,100], there is a sharp change in the service demand. The systems where agents use `Utility` were completely adapted to the previous service demand. As a consequence, there is a jump in the number of structural changes and in the average number of steps (see Figure 7(c)). Nevertheless, the average number of steps is lower than in systems that use `WPL`. There is an important drop in the success and in the efficiency of the system (see Figures 7(b) and 7(d)). Both algorithms need time to be able to start improving the structure of the network. At the end of this interval, both algorithms have improved the success rate of the solved queries and the mean path length is reduced (see Figure 7(a)). Nevertheless, in the case of `Utility`, the efficiency of the system is still better than the efficiency of `WPL` (see Figure 7(d)). Finally, as Figure 7 shows, in the third interval, the number of structural changes using `WPL` are enough to adapt to the current service demand (i.e., the service demand distribution of the first interval) since the adaptation in the second interval has only been partial. In the case of `Utility`, the network structural relations were adapted to the previous demand and initially the system requires a higher number of structural changes to adapt to the new service demand. Nevertheless, since the average number of steps remains low and the success rate is high, the efficiency is maintained as in previous intervals.

7. CONCLUSIONS

In this work, we have proposed a Service-Oriented MAS where agents offer their functionality through services. In contrast to other approaches in the literature which make use of the hierarchy of the entities of the system, in our system, all the agents are equal and only have a local view for performing self-organization actions. Another difference with current approaches of decentralized service discovery that start from a random structure for the system, we present an initial self-organized network structure that is based on a social feature called homophily. Therefore, when agents enter the system, they establish relations with other agents taking into account *homophily*, which is based on the semantic similarity of the services provided by the agents. The resultant *structure* is a self-organized growing network.

Agents in the system need to locate other agents that offer certain services in order to fulfill their goals. We have described a *decentralized service discovery process* through which, by considering homophily and the degree of connection with their direct neighbors, agents are able to reach the agent that provides the required service in just a few steps.

However, if service demand changes and the new service demand distribution does not correspond to the distribution of services among agents, the performance of the

service discovery could be affected. In order to adapt the structure of the system to changes in service demand, we have included a *self-organization* mechanism in the service discovery process. This mechanism exchanges current relations of agents that are not being used for new relations that are expected to be frequently used. During the service discovery, agents evaluate the utility of their current links and the suitability of their acquaintances. Not only does this utility take into account the traffic that passes through an agent, it also takes into account the type of services that the agent offers. Based on this information, each agent is able to decide when it is worthwhile to modify its structural relations with its current neighbors and then select the most appropriate acquaintances to replace these neighbors in order to maintain its degree of connection in the network. The set of acquaintances is not composed of agents that are randomly selected is the case in other approaches in the literature. In our proposal, the set of acquaintances is composed of agents that are found as a result of a service discovery process.

We performed several experiments to evaluate the effects of the inclusion of the proposed self-organizing mechanism in the service discovery performance. First, we analyzed the influence of a set of configuration parameters in our self-organization mechanism. Second, we compared our proposal with three different ones: (i) systems where service discovery does not include a self-organization mechanism; (ii) systems that have a structure that is completely adapted to the service demand; and (iii) systems that include a self-organization mechanism based on reinforcement learning. Finally, we evaluated the proposed self-organization mechanism in a dynamic environment with different service demands. In general, the inclusion of the proposed self-organization mechanism improved the efficiency of the service discovery process by reducing the number of steps needed to locate the required service and by increasing the number of successful searches. The rate of structural changes was reduced significantly as the system was getting adapted to the service demand. Furthermore, this mechanism performed well under situations where drastic changes in the service demand occured. As future work, we plan to extend this proposal to include aspects that are related to non-functional parameters of services and to take into account the of heterogeneous behavior of agents.

Acknowledgements. This work is partially supported by the Spanish Ministry of Science and Innovation through grants TIN2009-13839-C03-01, TIN2009-13839-C03-02 (co-funded by Plan E), CSD2007-0022 (CONSOLIDER-INGENIO 2010), TIN2012-36586-C03-01, and PROMETEOII/2013/019.

REFERENCES

S. Abdallah and V. Lesser. 2007. Multiagent reinforcement learning and self-organization in a network of agents. *Proc. of the 6th International Conference on Autonomous Agents and Multiagent Systems* (2007), 172–179.

L. A. Adamic and B. A. Huberman. 2002. Zipf's Law and the Internet. *Glottometrics* 3 (2002), 143–150.

Muntasir Al-Asfoor, Brendan Neville, and Maria Fasli. 2012. Heuristic Resource Search in a Self-Organised Distributed Multi Agent System. *Proc. of the 6th Internationl Workshop on Self-Organizing Systems* 7166 (2012), 84–89.

Mathieu Aquin, Salman Elahi, and Enrico Motta. 2010. Personal Monitoring of Web Information Exchange : Towards Web Lifelogging. *Proc. of the Web Science Conf.* (2010).

Ulrich Basters and Matthias Klusch. 2006. RS2D: Fast Adaptive Search for Semantic Web Services in Unstructured P2P Networks. *Proc. of the International Semantic Web Conference* 4273 (2006), 87–100.

Umesh Bellur and Roshan Kulkarni. 2007. Improved Matchmaking Algorithm for Semantic Web Services Based on Bipartite Graph Matching. *Proc. of the International Semantic Web Conference* (2007), 86–93.

Devis Bianchini, Valeria De Antonellis, and Michele Melchiori. 2009. Service-Based Semantic Search in P2P Systems. *Proc. of the European Conference on Web Services* (2009), 7–16.

Bartosz Biskupski, Jim Dowling, and Jan Sacha. 2007. Properties and mechanisms of self-organizing MANET and P2P systems. *ACM Trans. Auton. Adapt. Syst.* 2 (2007), 1–34. Issue 1.

A. Blanc, Yi-Kai Liu, and A. Vahdat. 2005. Designing incentives for peer-to-peer routing. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, Vol. 1. 374–385 vol. 1.

Michael Bowling and Manuela Veloso. 2002. Multiagent Learning Using a Variable Learning Rate. *Artificial Intelligence* 136 (2002), 215–250.

Frances M. T. Brazier, Jeffrey O. Kephart, H. Van Dyke Parunak, and Michael N. Huhns. 2009. Agents and Service-Oriented Computing for Autonomic Computing: A Research Agenda. *IEEE Internet Computing* 13, 3 (May 2009), 82–87.

Gianni Di Caro, Frederick Ducatelle, and Luca Maria Gambardella. 2005. AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications* 16 (2005), 443–455.

Tyson Condie, Sepandar D. Kamvar, and Hector Garcia-Molina. 2004. Adaptive Peer-to-Peer Topologies. *IEEE P2P Computing,* 0 (2004), 53–62.

Arturo Crespo and Hector Garcia-Molina. 2002. Routing Indices For Peer-to-Peer Systems. *Proc. of the 22nd International Conference on Distributed Computing Systems* (2002), 23–32.

E. Del Val, M. Rebollo, and V. Botti. 2012a. Enhancing decentralized service discovery in open service-oriented multi-agent systems. *Autonomous Agents and Multi-Agent Systems* (2012), 1–30.

E. Del Val, M. Rebollo, and V. Botti. 2012b. Promoting Cooperation in Service-Oriented MAS through Social Plasticity and Incentives. *Journal of Systems and Software* (2012), 520–537.

E. Del Val, M. Rebollo, and V. Botti. 2012c. Self-Organized Service Management in Social Systems. *Proc. of the 45th Hawaii International Conference on System Sciences* (2012), 810–817.

Ding Ding, Lei Liu, and Hartmut Schmeck. 2010. Service Discovery in Self-Organizing Service-Oriented Environments. *Proc. of the 2010 IEEE Asia-Pacific Services Computing Conference* (2010), 717–724.

S. N. Dorogovtsev and J. F. F. Mendes. 2003. *Evolution of Networks: From Biological Nets to the Internet and WWW*.

Giovanna (ed.) di Marzo Serugendo, Marie-Pierre (ed.) Gleizes, and Anthony (ed.) Karageorgos. 2011. *Self-organizing software. From natural to artificial adaptation.* Natural Computing Series.

Erik Einhorn and Andreas Mitschele-Thiel. 2008. RLTE: Reinforcement Learning for Traffic-Engineering. *Proc. of the 2nd International Conference on Autonomous Infrastructure, Management and Security* (2008), 120–133.

Jose Luis Fernandez-Marquez, Josep Lluis Arcos, and Giovanna Di Marzo Serugendo. 2012. A Decentralized Approach for Detecting Dynamically Changing Diffuse Event Sources in Noisy WSN Environments. *Applied Artificial Intelligence* 26, 4 (2012), 376–397. DOI:http://dx.doi.org/10.1080/08839514.2012.653659

Nelson Fernndez, Carlos Maldonado, and Carlos Gershenson. 2014. Information Measures of Complexity, Emergence, Self-organization, Homeostasis, and Autopoiesis. In *Guided Self-Organization: Inception*, Mikhail Prokopenko (Ed.). Emergence, Complexity and Computation, Vol. 9. Springer Berlin Heidelberg, 19–51. DOI:http://dx.doi.org/10.1007/978-3-642-53734-9_2

A. Forestiero, C. Mastroianni, and M. Meo. 2009. Self-Chord: A Bio-inspired Algorithm for Structured P2P Systems. *Proc. of the 9th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing* (2009), 44–51.

M. E. Gaston and M. desJardins. 2005. Agent-Organized Networks for Multi-Agent Production and Exchange. *Proc. of the 20th AAAI Conference on Artificial Intelligence* (2005), 77–82.

Nathan Griffiths and Michael Luck. 2010. Changing neighbours: improving tag-based cooperation. *Proc. of the 9th International Conference on Autonomous Agents and Multiagent Systems* (2010), 249–256.

P. Haase, R. Siebes, and F. van Harmelen. 2008. Expertise-based peer selection in Peer-to-Peer networks. *Knowledge and Information Systems* 15, 1 (2008), 75–107.

Philip N. Howard, Lee Rainee, and Steve Jones. 2001. Days and nights on the Internet. *American Behavioural Scientist* 45 (2001), 383–404.

Bernardo A. Huberman and Lada A. Adamic. 2000. The Nature of Markets in the WWW. *Quarterly Journal of Electronic Commerce* 1 (2000), 5–12.

Michael N. Huhns. 2002. Agents as Web Services. *IEEE Internet Computing* 6, 4 (July 2002), 93–95.

Michael N. Huhns and et al. 2005. Research Directions for Service-Oriented Multiagent Systems. *IEEE Internet Computing* 9, 6 (2005), 65–70.

Tomoko ITAO, Tatsuya Suda, Tetsuya Nakamura, Miyuki Imada, Masato Matsuo, and Tomonori Aoyama. 2001. Jack-in-the-Net: Adaptive Networking Architecture for Service Emergence. In *Proceedings of the Asian-Pacific Conference on Communications*. 9.

Emily M. Jin, Michelle Girvan, and M. E. J. Newman. 2001. Structure of growing social networks. *Phys. Rev. E* 64 (Sep 2001), 046132. Issue 4.

Sachin Kamboj and Keith S. Decker. 2007. Organizational self-design in semi-dynamic environments. *Proc. of the 6th International Joint Conference on Autonomous agents and multiagent systems*, Article 202 (2007), 335 - 337 pages.

M. Rahamatullah Khondoker, S. M. Taslim Arif, Nathan Kerr, and Dennis Schwerdel. 2011. Self-Organizing Communication Services in Future Network Architectures. *Proc. of the 5th International Workshop on Self-Organizing Systems* (2011).

Matthias Klusch, Benedikt Fries, and Katia Sycara. 2009. OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services. *Web Semantics Science Services and Agents on the World Wide Web* 7, 2 (2009), 121–133.

Dionisis Kontominas, Paraskevi Raftopoulou, Christos Tryfonopoulos, and EuripidesG.M. Petrakis. 2013. DS4: A Distributed Social and Semantic Search System. *Advances in Information Retrieval* 7814 (2013), 832–836.

Ramachandra Kota, Nicholas Gibbins, and Nicholas R. Jennings. 2012. Decentralized approaches for self-adaptation in agent organizations. *ACM Trans. Auton. Adapt. Syst.* 7, 1, Article 1 (may 2012), 28 pages.

P Lazarsfeld. 1954. Friendship as a Social Process: A Substantive and Methodological Analysis. *Freedom and Control in Modern Society* (1954).

Paulo Leito. 2013. Towards Self-organized Service-Oriented Multi-agent Systems. In *Service Orientation in Holonic and Multi Agent Manufacturing and Robotics*. Studies in Computational Intelligence, Vol. 472. 41–56.

W.S. Lin, H.V. Zhao, and K.J.R. Liu. 2009. Incentive Cooperation Strategies for Peer-to-Peer Live Multimedia Streaming Social Networks. *IEEE Transactions on Multimedia* 11, 3 (april 2009), 396–412.

Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng. 2001. Semantic Web Services. *IEEE Intelligent Systems* 16 (March 2001), 46–53. Issue 2.

M McPherson, L Smith-Lovin, and J Cook. 2001. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology* 27 (2001), 415–444.

Vivek Nallur and Rami Bahsoon. 2012. A Decentralized Self-Adaptation Mechanism for Service-Based Applications in the Cloud. *IEEE Transactions on Software Engineering* 99 (2012), 591 – 612.

A. Ouksel, Y. Babad, and T. Tesch. 2004. Matchmaking Software Agents in B2B Markets. *Proc. of the 37th Annual Hawaii International Conference on System Sciences* (2004), 1–9.

Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia P. Sycara. 2002. Semantic Matching of Web Services Capabilities. *Proc. of the 1st International Semantic Web Conference* (2002), 333–347.

Leonid Peshkin and Virginia Savova. 2002. Reinforcement Learning for Adaptive Routing. In *Proc. of IJCNN*. 1825–1830.

Paraskevi Raftopoulou and Euripides G. M. Petrakis. 2008. iCluster: a self-organizing overlay network for P2P information retrieval. In *Proc. of the 30th ECIR*. 65–76.

Sharmila Savarimuthu, Maryam Purvis, Martin Purvis, and BastinTonyRoy Savarimuthu. 2011. Mechanisms for the Self-organization of Peer Groups in Agent Societies. *Multi-Agent-Based Simulation XI* 6532 (2011), 93–107.

G. Di Marzo Serugendo, M. P. Gleizes, and A. Karageorgos. 2005. Self-organization in multi-agent systems. *KER* 20 (2005), 165–189.

AbdulKhalique Shaikh, SaadatM. Alhashmi, and Rajendran Parthiban. 2012. A Semantic Impact in Decentralized Resource Discovery Mechanism for Grid Computing Environments. *Proc. 12th International Conference on Algorithms and Architectures for Parallel Processing* 7440 (2012), 206–216.

Qixiang Sun and Hector Garcia-Molina. 2004. SLIC: A Selfish Link-Based Incentive Mechanism for Unstructured Peer-to-Peer Networks. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*. IEEE Computer Society, Washington, DC, USA, 506–515.

Mirko Viroli and Franco Zambonelli. 2010. A biochemical approach to adaptive service ecosystems. *Information Sciences* 180, 10 (2010), 1876 – 1892. DOI:http://dx.doi.org/10.1016/j.ins.2009.11.021 Special Issue on Intelligent Distributed Information Systems.

Li Wang. 2011. SoFA: An expert-driven, self-organization peer-to-peer semantic communities for network resource management. *Expert Syst. Appl.* 38, 1 (Jan. 2011), 94–105.

K Werbach. 2000. Syndication–the emerging model for business in the Internet era. *Harv Bus Rev* 78, 3 (2000), 84–93, 214.

Tom Wolf and Tom Holvoet. 2005. Emergence Versus Self-Organisation: Different Concepts but Promising When Combined. In *Engineering Self-Organising Systems*, SvenA. Brueckner, Giovanna Marzo Seru-

gendo, Anthony Karageorgos, and Radhika Nagpal (Eds.). Lecture Notes in Computer Science, Vol. 3464. Springer Berlin Heidelberg, 1–15.

Haizheng Zhang, W. Bruce Croft, Brian Levine, and Victor Lesser. 2004. A Multi-Agent Approach for Peer-to-Peer Based Information Retrieval System. *Proc. of the 3rd International Conference on Autonomous Agents and Multiagent Systems - Volume 1* (2004), 456–463.

Ming Zhong. 2006. Popularity-Biased Random Walks for Peer-to-Peer Search Under the Square-Root Principle. *Proc. of the 5th International workshop on Peer-To-Peer Systems* (2006).

Received February 2007; revised March 2009; accepted June 2009