Decentralized Approaches for Self-Adaptation in Agent Organizations

RAMACHANDRA KOTA, NICHOLAS GIBBINS,
and NICHOLAS R. JENNINGS, University of Southampton

Self-organizing multi-agent systems provide a suitable paradigm for developing autonomic computing systems that manage themselves. Towards this goal, we demonstrate a robust, decentralized approach for structural adaptation in explicitly modeled problem solving agent organizations. Based on self-organization principles, our method enables the autonomous agents to modify their structural relations to achieve a better allocation of tasks in a simulated task-solving environment. Specifically, the agents reason about when and how to adapt using only their history of interactions as guidance. We empirically show that, in a wide range of closed, open, static, and dynamic scenarios, the performance of organizations using our method is close ($70 - 90\%$) to that of an idealized centralized allocation method and is considerably better ($10 - 60\%$) than the current state-of-the-art decentralized approaches.

Categories and Subject Descriptors: I.2.11 [**Computing Methodologies**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Autonomic computing, self-organization, adaptation, organization structure, agent organization

ACM Reference Format:
Kota, R., Gibbins, N., and Jennings, N. R. 2012. Decentralized approaches for self-adaptation in agent organizations. ACM Trans. Autonom. Adapt. Syst. 7, 1, Article 1 (April 2012), 28 pages.
DOI = 10.1145/2168260.2168261 http://doi.acm.org/10.1145/2168260.2168261

1. INTRODUCTION

Autonomic systems that are capable of self-management have been advocated as a solution to the problem of maintaining modern, large, and complex computing systems [Kephart and Chess 2003]. Within this context, we contend that self-organizing multi-agent systems provide a suitable paradigm to develop these autonomic systems, because such self-organizing systems can arrange and rearrange their structure autonomously, without any external control, in order to adapt to changing requirements and environmental conditions. Furthermore, such adaptation needs to be performed in a decentralized fashion, so that the ensuing system is robust against failures; again, a characteristic that fits with the multi-agent paradigm [Tesauro et al. 2004]. With this motivation, this article explores the area of self-organization in systems of autonomous agents, and particularly focuses on adaptation of the structure in agent organizations.

This article is a significantly extended version of a previous paper Kota et al. [2009] published in *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'09)*, 10–15.
Authors' addresses: R. Kota (corresponding author), N. Gibbins, and N. R. Jennings, IAM Research Group, School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK; email: rck05r@ecs.soton.ac.uk.

DOI 10.1145/2168260.2168261 http://doi.acm.org/10.1145/2168260.2168261

In more detail, self-organization is here viewed as the "mechanism or the process enabling the system to change its organization without explicit external command during its execution time" [Di Marzo Serugendo et al. 2005]. In particular, any self-organizing system is expected to have the following properties: (i) no external control (all of the adaptation process is initiated internally and only changes the internal state of the system); (ii) dynamic operation (the system is expected to evolve with time; self-organization is a continuous process); and (iii) no central control (the organization is maintained only through local interactions of the individual components with no central [internal or external to the system] guidance). Building on this, we argue that the presence of appropriate self-organization principles in a distributed computing system can make the system autonomic. In this vein, De Wolf and Holvoet [2003] recommend that agent-based modeling is best suited to build such systems. Here, we are primarily interested in multi-agent systems that act as cooperative problem-solving organizations (i.e., those comprising cooperative autonomous agents that receive inputs, perform tasks, and return results)[1] because they clearly act as an abstract model of the distributed systems. Hence, we focus on developing self-organization techniques for such agent organizations. Moreover, we believe decentralized structural adaptation is the most appropriate way of achieving self-organization in agent organizations. Here, the structure of an organization is a manifestation of the relations between the agents, which, in turn, determine their interactions. Consequently, adapting the structure involves changing the agent relations, and thereby redirecting their interactions.

To make it clear, consider a sample scenario of the interconnected network of a university as a form of autonomic grid computing system. Being a university, it contains various labs with their own specialized computing systems, as part of the overlaying network of the university. For example, a computer in the geography lab might contain specialized software for analysing GIS maps, while that in a graphics lab can render high-quality images. Now, these computers providing different services will need to interact with each other to perform complex tasks (say, creating detailed city maps by analyzing GIS data). Moreover, as these individual computers are controlled by different people in different labs, the respective loads on them, at any time, cannot be known or predicted. Also, some might go offline when they are disconnected, some might be upgraded, and so on. Hence, the computers need to continuously adapt their interactions with others in the university network to keep up with the changes and, at the same time, optimize the overall performance.

To illustrate further, let us focus on only a few computers in the GIS labs and the graphics labs, as depicted in Figure 1(a). Initially, computer Y_2 is working on some project involving the city Seoul, whose GIS information is present in X_3. Thus, Y_2 maintains relations with Y_1 and X_3. Similarly, Y_1 and X_1 have a relation and so on. However, X_3 was switched off by its owner when she went on vacation, as in Figure 1(b). Then, Y_2, left with no other resort, starts enquiring for its GIS information from Y_1 who then redirects the queries to its relation X_1 and sends back the information to Y_2. In such circumstances, Y_2 and X_1 should realize this and start maintaining a relation directly between them to reduce both the computation load and memory usage on Y_1, also saving bandwidth and resulting in a faster passage of the information considering that the GIS data, which tends to be huge, need not be copied to Y_1 in between (see

[1]The problem solving part of this definition is in contrast to organizations that just provide guidelines to be obeyed by agents participating in the organization to achieve their individual goals (see, for example, Sierra et al. [2004]). Specifically, these organizations do not have any particular goals to achieve, but only act as regulating authorities. Thus, they do not look to accomplish any defined tasks, and cannot be mapped onto distributed computing systems. The cooperative part of the definition is in contrast to those comprising self-interested and often competing agents, like in virtual organizations [Norman et al. 2004].

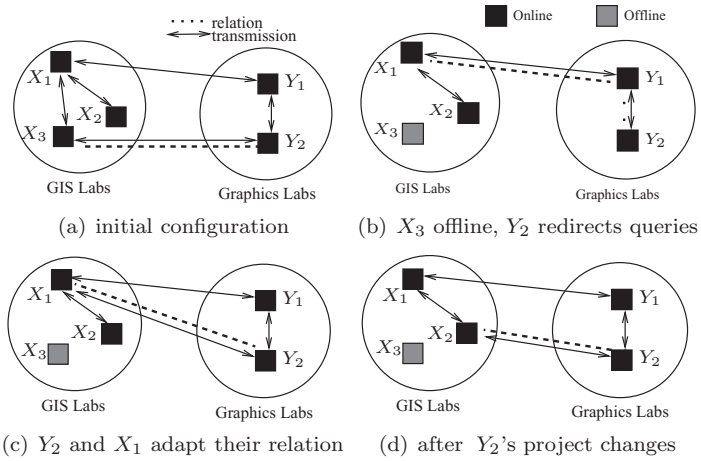


Fig. 1. An example of structural adaptation.

Figure 1(c)). With time, that Seoul-based project comes to an end and Y_1 is then being used for a new project relating to *SaoPaolo*. This GIS information is not present with X_1, but resides with X_2. Thus, instead of interacting indirectly via X_1, Y_2 and X_2 should then form a direct relation. At the same time, Y_2 and X_1 should realize that their interactions are not frequent anymore and dissolve their relation, as shown in Figure 1(d). Thus, structural adaptation is especially critical in situations where it can help the organization cope with both internal changes and those in the external environment.

In this context, Mathieu et al. [2002] suggest that an adaptation method is important to improve the performance (in terms of costs and task completion times) of organizations, though they do not actually provide such a method. Furthermore, autonomic systems are expected to be deployed in uncertain and changing environments where neither the components, their characteristics, nor the tasks facing the system will remain constant. In more detail, the system will be expected to continue performing well in scenarios where agents might be added or removed from the organization, the properties of the existing agents might be changed with time (they might start providing new services, lose services, or gain more resources), and similarly, the characteristics of the task stream (the type and rate of tasks) might also vary with time. In such cases, structural adaptation will enable the agents to reorganize their interactions to better suit the changed circumstances. Thus, structural adaptation is especially critical in such situations as it can help the organization cope with both internal changes and those in the external environment.

Moreover, as the adaptation process itself will require some computation, metareasoning is also needed by the agents to decide "whether to adapt" (in addition to "how to adapt") or to continue performing the tasks without adaptation. As a sample scenario, consider our earlier example. Y_2 has limited computational resources (processor cycles and memory) available to it. Given that it has to process a continuous stream of tasks for its projects, it has to make the best possible use of the resources for a good performance (in terms of tasks completed for the project). In addition to those computational tasks, we have seen that Y_2 also needs to maintain the best set of relations to help in its task allocation. This evaluation and modification of relations (structural adaptation) by Y_2 takes up its resources as well. Y_2 will have to balance its limited resources between doing its actual tasks and this adaptation reasoning. Therefore, it

becomes imperative for Y_2 to choose smartly between when to evaluate the structure for adaptation and when to continue with the current structure (that is, the current relations) without evaluation, thereby needing metareasoning. Now, such metareasoning in a multi-agent systems context has been shown particularly important for resource-bounded agents in uncertain environments [Raja and Lesser 2004]. Thus, we believe it is an important issue that needs to be addressed in our context because we also deal with agents adapting in the face of limited computational resources and present in dynamic environments where the tasks and agent properties are unpredictably changing with time.

Against this background, we present a novel structural adaptation method for problem-solving agent organizations. Following self-organization principles, the method is a decentralized and continuous process that is followed by every agent to decide on when and how to adapt its relations, based only on locally available information. Moreover, the adaptation method only involves changing the structural relations between the agents and does not need the agents to change their internal properties like services/skills or capacities and neither does it need new agents to be added, or existing agents to be removed from the system. Therefore, it can even be applied to scenarios where such modifications to the agents are not permitted to the adaptation mechanism. Thus, our mechanism can serve as a self-management tool similar to those envisioned in autonomic systems.

Our adaptation method enables pairs of agents to continuously and locally reevaluate their inter-relations on the basis of their past interactions. Using the method, every pair of agents can calculate the utility of the possible relations between them and choose the most beneficial one. Additionally, the agents are also able to decide when to initiate such calculation and with which other agents. Furthermore, the organizations can be open and dynamic as well. In such systems, agents might be entering or leaving the system and/or their properties changing with time, thereby representing distributed systems in which resources are added, removed, updated, or changed as time goes on. In this context, our method also aids the agents to adapt in these *open organizations*. Using simple principles based on the current context of the existing agents, newer agents are easily assimilated into the structure by the method. Similarly, agents in *dynamic organizations* are able to adapt the structure to the quickly changing circumstances by associating time-decaying weights to the past interactions while calculating utilities. Here, it is important to note that our focus is *not* on distributed task allocation. Rather, it is on the underlying structure that the agents use while allocating and executing tasks. In this way, our work enables the agents to adapt their structure towards optimizing the efficiency of task completion, and is independent of the actual task allocation algorithms that the agents might employ.

In summary, our method can be seen to extend the state-of-the-art in terms of structural adaptation mechanisms for agent organizations by being the first that is generically applicable to models with a broad range of inter-agent relations and by addressing the metareasoning aspects of adaptation in a completely decentralized fashion. It is also the first adaptation method that is suitable for open and dynamic organizations where the agents and their internal characteristics are changing with time. Finally, we advance the state-of-the-art by providing a self-organization inspired approach for decentralized adaptation in formally specified organizations (as opposed to structureless systems like swarms and ant colonies).

In the next section, we review the existing literature relevant to the problem at hand. We follow it with a description of the model of a problem-solving agent organization that will act as the abstract platform on which to base our adaptation mechanism (Section 3). By using such a generic platform, instead of focusing on a particular existing system, we can develop a general method that can be applied to a wide variety of applications.

We first present the fundamentals of our adaptation process in Section 4 and then show how to extend it for open and dynamic scenarios. Then, we demonstrate the effectiveness of our approach through experimental evaluation in Section 5. Finally, we conclude in Section 6.

2. RELATED WORK

Self-organization can be generated in multi-agent systems in several ways [Di Marzo Serugendo et al. 2006; Bernon et al. 2006]. For example, it may emerge from stigmergic (indirect coordination through traces left in the environment) or reinforcement mechanisms in agents [Mano et al. 2006] or it can arise from the locally cooperative actions of the agents [Capera et al. 2003a; 2003b]. To date, however, most of the self-organization mechanisms are not applicable to an explicitly modeled agent organization because, being based on reactive agents interacting in unstructured ways, they cannot easily be incorporated into agents that are working towards organizational goals. The few mechanisms that do consider agent organizations are usually centralized in nature, requiring a few specialized agents to manage the adaptation process for all the agents. For example, Hubner et al. [2004] present a "controlled reorganization" mechanism which is a top-down approach in which a specialized group of agents perform the reorganization process for the whole organisation. Therefore, it is neither decentralized nor continuous. Similarly, Bou et al. [2006] present a centralized reorganization mechanism in which the central authority named "autonomic electronic institution" modifies the norms of the institution to achieve institutional goals. Thus, this work does not focus on the organization structure. A centralized mechanism that involves the organization structure is presented by Hoogendoorn [2007] where a maxflow network-based approach is used to identify bottlenecks in the organization, the corresponding agents or nodes are replicated, and new structural links are added to connect them to the organization. It also aims to improve the capacity of the organization by adding links and nodes, but does not attempt to optimize by removing redundant links or nodes.

In contrast, Horling et al. [2001] suggest a somewhat distributed method (using a central blackboard). However, their approach involves a diagnostic subsystem for detecting faults in the organization that map to some fixed predesigned reorganization steps. But, such a method is not applicable when all the states of the system cannot be anticipated by the designer. Similar drawbacks also exist with the approach presented by Wang and Liang [2006] in which the transformation of organization structure occurs by agents shifting between roles on the basis of some predefined rules corresponding to different scenarios. Finally, a method called organization self-design [Ishida et al. 1992; Kamboj and Decker 2007] achieves self-organization by dynamically spawning and merging agents in response to the changing requirements. However, since agent-based development of autonomic systems involves modeling the individual components as agents, changing the characteristics of these components may not be possible on all occasions due to physical and accessibility limitations (e.g., data centers located in remote places cannot easily be replicated). Moreover we are interested in adapting the inter-agent interactions, rather than changing the agents internally (for the reasons detailed in Section 1).

A self-organization approach that has been successfully applied in a multi-agent system is demonstrated by Schlegel and Kowalczyk [2007]. They tackle the problem of resource allocation by proposing a distributed algorithm that does not require any central controller. Their agents attempt to optimize their task allocations to servers by forecasting the future task load on the servers on the basis of the history of server utilization, obtained from the completed tasks at those servers. On the basis of the forecasts on each of the servers, the agent chooses the server with the maximum capacity forecasted. In this way, efficient resource allocation emerges from the indirect

interactions between the agents (as the agents only interact with the servers). The major difference between their work and ours is that, in their work, the agents do not interact directly and take all decisions independently, while in our model, the agents need to interact with each other to collectively decide about their relations. Furthermore, in this case, the self-organization process influences the task allocations on a case-to-case basis, while we require self-organization at the higher level of agent relations that, in turn, influence the task allocations.

On similar lines, Gershenson [2007] demonstrates a self-organization approach for the problem of task assignment in agent networks. An agent, that receives a task, needs to send out some dependency requests to its neighboring agents. The self-organization process works by first identifying the agent (say a_x) with the longest queue. Then, among the agents dependent on a_x, the one with the largest waiting period (say a_y) chooses another agent (one with the shortest queue) (a_z) to replace a_x as its neighbor. Therefore, the global knowledge of the queues of every agent is required in this method, which is not always a valid assumption.

Additionally, there has been some work using self-organization in agent systems using a "holonic" architecture. The methods are based on *holarchies*: hierarchies made up of holons. Holons are entities that can exist independently or can join with other holons to form bigger holons. Holons dynamically altering the holarchy, according to changes in the environment, form the basis of self-organization [Bongaerts 1998; Fischer 2005]. For example, Hilaire et al. [2008] use a holonic architecture for decentralized decision making in the agent system. However, such holarchy-based approaches require a strict hierarchy between the groups of holons. Also, while this approach helps the agents in decision making regarding the tasks, it does not assist with reasoning about the structure itself.

As stated in Section 1, we seek a method that enables the agents to self-organize in distributed systems that have to operate in highly dynamic environments. Now, such an approach is presented by Forestiero et al. [2008] for information dissemination in a dynamic grid computing system. In their case, agents travel through the grid replicating information and discovering new resources based on some biology-inspired algorithms. However, their method is specifically applicable to resource discovery and update only, while we seek a self-organization approach for the very different problem of structural adaptation. Nevertheless, the usefulness of a self-organization mechanism in a dynamic environment is amply demonstrated by their work. More specifically, we seek a mechanism that will enable the agents to locally adapt the structure in a dynamic environment. Such methods are generally developed for peer-to-peer systems or networks. To this end, Biskupski et al. [2007] survey the existing self-organizing methods for such systems by comparing them against their model of agent-based self-organization. Specifically, their localized mechanisms incorporate concepts of feedback, local evaluation functions, and decay. Although our domain is more complex, as it deals with agent organizations (rather than networks) which contain several possible types of relations or links between agents influencing both task allocation and load balancing in the organization, the ideas of feedback, decay, and local evaluation functions are useful to us too. Thus, we will be including these basic ideas into the design of our approach.

Just like the P2P networks, social networks [Watts 2001; Jackson and Watts 2002] also provide a suitable domain to investigate structural adaptation methods. In this vein, Gaston and desJardins [2005] focus on agent networks, which comprise a set of agents with some undirected acquaintance links between them. Their work deals with agent-based rewiring of the links in order to improve dynamic team formation, thus somewhat resembling structural adaptation in organizations. However, their model assumes that only one type of relation exists in the system, and that the number of

relations possessed by an agent has no effect on its computational resources. Under these assumptions, their methods always result in scale-free network structures, which are unrealistic in cases when agents have to expend resources for managing and delegating tasks based on their relations. Glinton et al. [2008] improved over this approach by limiting the number of links at an agent and using a token-based adaptation approach for a better spread of links across the network. As they state, their algorithm "randomly walks a token from an agent wanting to change its links to an agent to which a link would be potentially beneficial". However, the model was still restricted to single link-type,[2] ignored load due to the existing links, and also the metareasoning aspects of adaptation. Nevertheless, as this is the latest and also the most relevant work that comes close to our requirements, we have implemented this and compared its performance against ours. Another work on similar lines, by Abdallah and Lesser [2007], deals with task allocation in agent networks. They use the "Weighted Policy Learner" algorithm for learning task allocation where agents are connected to each other via a network. These agents use the information gained from the learning mechanism to guide each other into changing their set of neighbors, thereby reorganizing the network. However, as in the aforesaid work, their network supports only one type of link. Moreover, every type of task has its own separate agent network. Hence, the method is not required to adapt the same network when faced with various kinds of tasks. Also, since the number of incoming or outgoing links of an agent is assumed to not affect its resources, an agent is able to form a link to another agent without requiring the consent of the other agent. Such an assumption is not always valid and more generic organization models would require that two agents agree on the relation between them. Finally, the fact that an agent's resources might be expended by the reorganization process is also ignored.

Finally, as suggested in Section 1, if the adaptation process also requires computation, metareasoning is needed by the agents to decide whether and when to adapt. Metareasoning, in general, has been explored in a multi-agent systems context [Alexander et al. 2007; Hogg and Jennings 2001], but has not previously been applied to self-organization scenarios. In particular, Conitzer [2008] emphasises that generic metareasoning problems tend to be computationally hard and it is more productive to focus on individually solving the particular cases where metareasoning is required. With this knowledge, we seek to only solve our particular metareasoning problem and thereby our approach may or may not be applicable to other metareasoning scenarios.

3. THE ORGANIZATION MODEL

In this section, we present our organization framework by first describing the task model and a basic organization representation. Next, we discuss the modeling of open and dynamic organization before detailing the mechanism for measuring an organization's performance.

Organization modeling involves modeling the agents comprising the organization, the organizational characteristics, and the task environment. There are several existing frameworks for such modeling in the literature [Dignum 2003; Vazquez-Salceda et al. 2005; Sierra et al. 2004; Deloach et al. 2008]. However, we mainly build on the ideas presented by Jin and Levitt [1996], Ferber and Gutknecht [1998], and Hannoun et al. [2000]. A preliminary version of the consolidated model was presented by Kota et al. [2008]. However, here we extend it by making the organization open and dynamic (see Section 3.2). Moreover, the evaluation mechanism (Section 3.3) has been made

[2]Having only a single link-type makes all inter-agent links homogeneous, thereby restricting the model by not allowing for any kind of classification of the links on the basis of any characteristics like say, amount of interaction or speed of interaction.

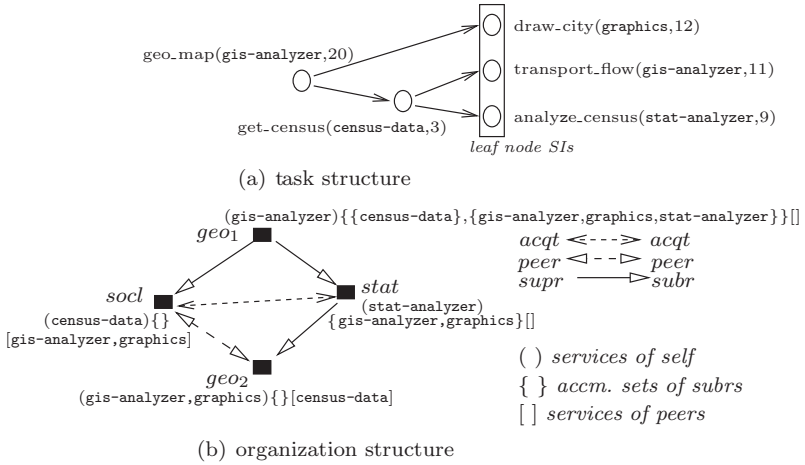


Fig. 2. Representation of an example task and organization.

more intuitive and realistic than before. In this context, it should be noted that our contribution is the adaptation method and not the organization model per se. Furthermore, although our adaptation method is demonstrated using this particular organization model, it can be equally well applied to other organization models, if desired.

3.1. Fundamentals of the Model

In our model, the agent organization comprises a group of problem solving, cooperative agents situated in a task environment. By problem solving, we mean agents receive certain input tasks to achieve, execute these tasks, and return the outcomes. Correspondingly, the task environment presents a continuous dynamic stream of tasks that have to be performed. In addition, the environment also has costs associated with passing messages between the agents (communication) and changing their relations (reorganization).

In more detail, the tasks are modeled as workflows composed of a set of Service Instances (SI), each of which specifies the particular service and the amount of computation required (defined in terms of the number of units that need to be available at the agent executing it). These SIs need to be executed following a precedence order which is specified in the form of dependency links between the SIs. This dependency structure is modeled as a tree. The execution of a task begins at the root node and the task is deemed complete when all its nodes have been executed, terminating at the leaf nodes. Figure 2(a) shows an example task composed of five SIs each requiring a particular service and a specified amount of computation. The required order of execution is shown by the dependency links between the SIs. That is, geo_map needs to be executed first, followed by its child nodes, draw_city and get_census (which are executed in any order or even in parallel).

The organization consists of a set of agents A that provide these services. Every agent is capable of a set of services and possesses a fixed computational capacity. Thus, an agent is of the form $a_x = \langle S_x, L_x \rangle$, where $S_x \subseteq S$ (S is the complete set of services provided by the organization) and L_x is the agent's capacity defined in terms of available computational units in a time step (these are consumed by the SIs as they are executed). Tasks enter the system at random time steps and their processing should start immediately. The processing of a task begins with the assignment of the root SI to some agent. The agent that executes a particular SI is, then, also responsible for

the allocation of the subsequent dependent SIs (as specified by the task structure) to agents capable of those services. Thus, the agents have to perform two kinds of actions: (i) execution and (ii) allocation. Consider an agent executing SI geo_map in Figure 2(a). After completing the execution, that agent needs to find and allocate appropriate agents to execute draw_city and get_census, the dependent SIs. Moreover, every action has a load associated with it. The load incurred for the execution of a SI is equal to the computational amount specified in its description, while the load due to allocation (called management load) depends on the relations of that agent (will be explained later). As every agent has a limited computational capacity, an agent will perform the required actions on a first-come first-served basis, in a single time step, as long as the cumulative load (for the time step) on the agent is less than its capacity. If the load reaches the capacity and there are actions still to be performed, these remaining actions will be deferred to the next time step, and so on.

As described earlier, agents need to interact with one another for the allocation of the SIs. These interactions are regulated by the structure of the organization. This structure is based on the relationships between the agents. The type of relationship can be categorized into different levels. We consider the following: (i) *stranger*, not aware of each other; (ii) *acquaintance*, knowing about the presence of, but no interaction; (iii) *peer*, low frequency of interaction; and (iv) *superior-subordinate*, high frequency of interaction. It is clear that a higher-level relation (like superior-subordinate) contains the properties of the lower level (like being acquaintances of each other) in addition to some more characteristics (like the superior knowing the services being provided by the subordinate and delegating SIs to it).[3] Therefore, the relation between any two agents can exist in only one of these states at a time. In particular, the type of relation present between two agents determines the information that they hold about each other and the interactions allowed between them. We can distinguish between the various relations as follows.

—An agent possesses information about the services that it provides, the services provided directly by its peers, and the accumulated service sets of its subordinates. The accumulated service set (*AccmSet*) of an agent is the union of its own service set and the accumulated service sets of its subordinates recursively. Thus, the agent is aware of the services that can be obtained from the subgraphs of agents rooted at each of its subordinates, though it might not know exactly which particular agent is capable of the service.
—During the allocation of an SI, an agent will always try to perform the SI by itself if it is capable of the service and has available computational capacity. Otherwise, it will delegate it to one of its subordinates (which contains the service in its accumulated service set). If there are no suitable subordinates (none of the subordinate subgraphs is capable of that service) and it is capable of the service itself (but did not initially assign to self because its capacity is filled), then it will add the SI to its task waiting queue for execution. However, if it is not capable of the service (and nor are its subordinates), it will try to delegate the SI to its peers. If it is unable to do so either (no peer is capable of the service) it will pass it back to one of its superiors (who will have to find some other subordinate or peer for delegation).

Therefore, an agent mostly delegates SIs to its subordinates and seldom to its peers. Thus, the structure of the organization influences the allocation of SIs among the agents. Moreover, the number of relations of an agent contributes to the management load that it incurs for its allocation actions, since an agent will have to sift through

[3]In this view, the stranger relation represents the fact that both agents belong to the organization (though they don't know each other), thus it is at a lower level than the acquaintance relation.

its relations while allocating an SI. Therefore, an agent with many relations will incur more management load per allocation action than an agent with fewer relations. Also, a subordinate will tend to cause management load more often than a peer because an agent will search its peers only after searching through its subordinates and not finding a capable agent. As all the agents in the organization are cooperative and work selflessly for the organization, an agent willingly accepts all SIs delegated by its superiors or peers. Note that, to avoid an infinite loop of delegation, the superior-subordinate (also called authority) relations are not permitted to have cycles. Also, the relations are mutual between the agents, that is for any relation existing between two agents, both the concerned agents respect it. In total, the authority relations impose the hierarchical structure in the organization while the peer relations enable the otherwise disconnected agents to interact closely. These types of relationships are sufficient to describe the different kinds of interactions possible in a task allocation setting. Moreover, having such multiple types of relations to select from (rather than just a single link-type) enables the agents to classify their links with the other agents, thereby providing more richness to the model and also helping them reason about their interactions more easily. Nevertheless, the types of relations described here are just canonical structures. For example, there could be an additional relation, say *manager*, which is like the authority relationship but in this, the superior is also aware of the current workload on the subordinates and thus can delegate SIs more wisely. Using our organization model, we also abstract away the complex interaction problems relating to issues like service negotiation, trust, and coordination. We do so to focus on the essence of self-organization and to isolate its impact on system performance.

Figure 2(b) shows an example organization. The services that an agent provides are shown beside it in parentheses. The services that an agent can seek from its subordinates, the *AccmSet*, is shown in curly braces, while the services that it can seek from its peers are shown in square brackets.

3.2. Open and Dynamic Organizations

Given an organization, the agents in it can remain unchanged over its existence, or they might change with time. For example, new agents might enter the organization and/or some existing agents might leave. In this way, the organization can be *open*. Moreover, even within a given agent, the properties can change with time. It might, for example, start providing new services and/or lose previous services. Thus, the organization can be *dynamic*. In contrast, organizations in which the set of agents is constant are called as *closed* and those in which the agent properties do not change are considered *static*. Thus an organization can be closed or open and, in addition, be static or dynamic. Up to now, our description of the organization model is sufficient to represent closed and static organizations. Thus, in the following, we discuss how to extend it to model open and dynamic organizations.

—*Open Organizations.* For this kind of organizations, the set of agents A changes with time. In particular, we focus on those organizations that have some permanent agents to begin with (similar to closed) and some temporary agents who join later, at specified "start-times," and also leave the organization at the expiration of their "life-times." We look at these types of open systems initially because in them, the service set S of an organization can be kept constant (the temporary agents will offer services chosen from the same S as the permanent ones). In this way, our method can focus solely on the changes to the overall capacity (resulting from the temporary agents) instead of the service discovery aspect that might have been needed. Consequently, these open organizations represent distributed systems in which additional resources might be added to tackle the workload and withdrawn later on (as discussed in Section 1).

—*Dynamic Organizations*. In these organizations, the properties of the agents are changing with time. As described earlier, in our model, an agent a_x has a service set S_x that it provides. Since $S_x \subseteq S$, the services $(s_i, s_j \dots)$ belonging to S_x can change with time. In particular, we look at scenarios, where the agents start with their respective service sets and then additional services (from S) are added to these sets with time (either gradually or suddenly). Similarly, we also look at scenarios where services are removed from the service sets of the agents with time. The way we generate these dynamic organizations is explained in detail in Section 5.1.

3.3. Organizational Performance Evaluation

The performance of a computing system denotes how well it performs its tasks. In terms of an agent organization, the performance measure can be abstracted to the profit obtained by it. In our model, the profit is simply the sum of the rewards gained from the completion of tasks when the incurred costs have been subtracted. In more detail, the cost of the organization is based on the amount of resources consumed by the agents, in addition to their computational capacities. In our case, this translates to the cost of sending messages (communication) and the cost of changing relations (reorganization) between the agents. Thus, the cost of the organization is

$$cost_{ORG} = C. \sum_{a_x \in A} c_x + D.d, \tag{1}$$

where C is the communication cost coefficient representing the cost of sending one message between two agents and D is the reorganization cost coefficient representing the cost of changing a relation. c_x is the number of messages sent by agent a_x and d is the number of relations changed in the organization.

As stated earlier, agents have limited capacities and their computational load cannot increase beyond this capacity. The load l_x on agent a_x in a given time step is

$$l_x = \sum_{si_i \in W_{x_E}} p_i + M \sum_{si_j \in W_{x_F}} m_{j,x} + R.r_x. \tag{2}$$

—p_i is the amount of computation expended by a_x for executing SI si_i (determined by the task description of the SI si_i);
—$m_{j,x}$ is the number of relations considered by a_x while allocating SI si_j;
—W_{x_E} is the set of SIs (possibly belonging to several tasks) being executed by a_x;
—W_{x_F} is the set of SIs being allocated by a_x;
—M is the "management load" coefficient denoting the computation expended by an agent to consider one of its relations while allocating a single SI;
—R is the "reorganization load" coefficient, denoting the amount of computational units consumed by an agent while reasoning about reorganization with another agent;
—r_x is the number of agents that a_x initiated reorganization deliberation in that time step.

In this way, M represents the computational complexity resulting from the relations of an agent. If an agent has more relations, it will need more computation for allocating an SI. The increase in computation caused by one relation at an agent is given by M. Depending on the domain, the value of M can be fixed and be the same for all agents, irrespective of their relations, or can be agent-specific and also vary depending on the number of relations at the agent (perhaps increasing exponentially as the number of relations increases). Similarly, R is used to represent the excess load caused by reasoning about reorganization. Since the load l_x of a_x cannot exceed its capacity L_x, any excess SIs will be waiting for their turn, thus delaying the completion time of the tasks.

The rewards obtained by the organization depend on the speed of completion of tasks. In particular, a task w completed on time accrues the maximum reward which is the sum of the computation amounts of all its SIs. For delayed tasks, this reward degrades linearly with the extra time taken until it reaches 0

$$reward_w = \sum_{i=0}^{|si_w|} p_i - \left(t_w^{taken} - t_w^{reqd}\right), \tag{3}$$

where si_w is its set of SIs, t_w^{taken} represents the actual time taken for completing the task, while t_w^{reqd} is the minimum time needed. Thus, the total reward obtained by the organization is the sum of the rewards of the individual tasks completed by the organization: $reward_{ORG} = \sum_{w \in W} reward_w$ where W is the set of all tasks. The organization's performance is measured by its profit.

$$profit_{ORG} = reward_{ORG} - cost_{ORG} \tag{4}$$

Thus, for higher profits, the reward should be maximized, while the cost needs to be minimized. It is important to note that the agents only have a local view of the organization. They are not aware of all the tasks coming in to the organization (only those SIs allocated to them and the immediately dependent SIs of those allocated SIs) and neither are they aware of the load on the other agents. In spite of this incomplete information, they need to cooperate with each other to maximize the organization profit by maintaining the most useful relations which lead to faster allocation and execution of tasks.

4. STRUCTURAL ADAPTATION

This section details our work on developing a self-organization-based structural adaptation method that can be employed continually by all the agents in a problem-solving organization. The aim of the adaptation method is to improve the performance of the organization. Our adaptation method is based on the agents forging and dissolving their relations with other agents, thereby modifying the organization structure. It uses only the history of agent interactions since we do not assume that agents possess any information about the tasks coming in the future. In this section, we first present the basics of the method and how it is to be applied by the agents in an organization. Following that, we show how to extend it to deal with open and dynamic organizations.

We present the fundamental mechanism, in a pseudocode form in Algorithm 1, for how an agent a_x should reorganize at a given time step. The first component (line 1) refers to the metareasoning aspect of choosing the particular acquaintances for initiating the reorganization process. The second component (lines 3–9) explains how it should adapt its relation with one such acquaintance a_y.

We formulate this part using a decision-theoretic approach since it provides us with a simple and suitable methodology for representing adaptation in terms of actions and utilities. We denote the actions available to a pair of agents as those changing the relation between them. Consequently, the set of actions available to a pair depends on their relation (line 3). In our model, for every pair of agents that are not strangers,[4] the relation between them has to be in one of the states: acquaintance relation, peer relation, or authority relation. For each of these states, there are three possible choices of action available to the agents as shown in Figure 3. For example, action 1(ii) ($\mathsf{form\_subr}_{x,y}$)

[4]The stranger relation is not considered as a state because if the pair of agents do not know each other, then they cannot seek to adapt their relation. Getting to know about strangers is an issue of service discovery and is out of the scope of this work. Similarly, two acquaintances can only become strangers by *forgetting* knowledge or memory of each other, again out of scope of this work.

ALGORITHM 1: Adaptation algorithm in terms of agent a_x

```
1 Chosen ← selected from the acquaintances set of a_x;
2 foreach a_y ∈ Chosen do
3 |   Actions ← possible_actions(x, y);
4 |   U_{x,y} ← ∅;
5 |   foreach e ∈ Actions do
6 |   |   U_e ← calculate_utility_{x,y}(e);
7 |   |   U_{x,y} ← U_{x,y} ⋃ U_e;
  |   end
8 |   e_best ← argMax(U_{x,y});
9 |   take action e_best with a_y;
  end
```

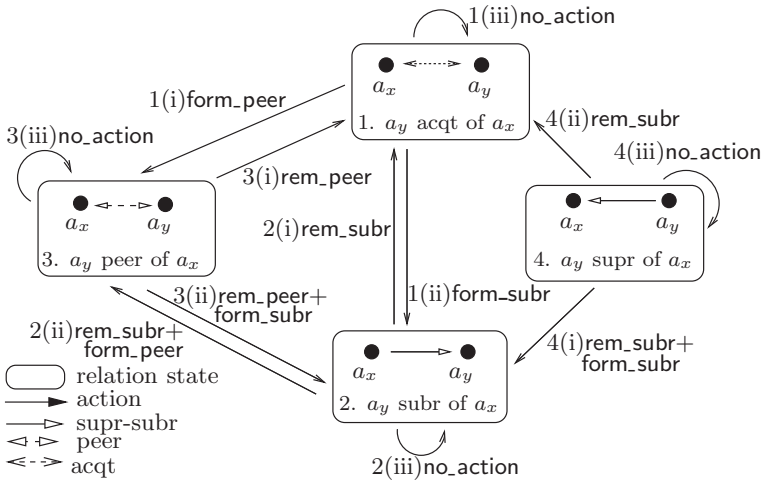


Fig. 3. State transition diagram.

denotes that a_x and a_y take the action of making a_y a subordinate of a_x and transition from state 1 to 2. A transition from state 2 to 4 is not needed because it is equivalent to the transition from 4 to 2, by interchanging a_x and a_y. Similarly, transitions from 1 to 4 or between 3 and 4[5] are not required. If there were more types of relations in the organization model, there would be correspondingly more states and transitions to represent them. However, the essence of the adaptation method will remain the same.

As can be seen, the transition actions are composed of four atomic types: form_peer, rem_peer, form_subr, and rem_subr, which translate to forging and dissolving the peer or authority relations (as agents are acquaintances of each other, by default). The actions are mutually exclusive and can be performed if the relation between the agents is in the requisite state (as explained later). Obviously, each of these actions has to be jointly performed by the two agents involved in changing the relation. Furthermore, the actions are deterministic (there is no uncertainty regarding the outcome of an action which is the formation or deletion of a link; only the utility of the outcome is not predetermined). The utility of performing an action (U_e in line 6) is given by value

[5]In state 4, when a_z is an indirect superior of a_x via a_y, it is not possible for a_x to have a_z as its subordinate (since cycles are not permitted). Hence, it dissolves its relation with its immediate superior in the authority chain a_y and goes to state 1 with respect to a_y and then forms a relation with a_z.

function V (also called an ordinal utility function) associated with the relation. Since our environment is characterized by various factors, like the costs and the load on the agents, V will have multiple attributes to represent them. In terms of two agents, a_x and a_y, the five attributes that will constitute V are: (i) change to the load on a_x, (ii) change to the load on a_y, (iii) change to the load on other agents of the organization, (iv) change to the communication cost, and (v) reorganization cost. Moreover, this set of attributes exhibits Mutual Preferential Independence (MPI). That is, while every attribute is important, it does not affect the way the rest of the attributes are compared. Therefore, V can be represented simply as a sum of the functions pertaining to the individual attributes.

$$V = \Delta load_x + \Delta load_y + \Delta load_{OA} + \Delta cost_{comm} + \Delta cost_{reorg} \tag{5}$$

In this way, depending on the state, the agents jointly calculate the expected utilities for the possible actions using the value function (which are stored in $U_{x,y}$ at line 7), and then choose the action giving the maximum expected utility (line 8). Being cooperative, the agents do not have conflicts as the value corresponds to the social utility of the relation to the organization and not to the individual agents. The evaluation for no_action will always be 0 as it does not result in any change to the expected load or cost of the organization. The evaluation for the rest of the actions is obtained from Eq. (5). In the case of the composite actions (like rem_subr+form_peer) the value will simply be the sum of the individual evaluations of the comprising actions. Moreover, since any action will be taken jointly by the two agents involved, even the evaluation is jointly conducted by the agents with each of them supplying those attribute values accessible to them.

To understand further, let us look at the utility calculation (Eq. (5)) for the action $\text{form\_subr}_{x,y}$ when a_x and a_y are just acquaintances (state 1). Here, $\Delta load_x$, representing the estimated change to the load on a_x, is calculated by considering that a new subordinate a_y will lead to an increase in the management load on a_x every time it tries to allocate an SI. This is quantified as

$$\Delta load_x = -M * Asg_{x,total} * filled_x\left(t_x^{total}\right)/t_x^{total}, \tag{6}$$

where $Asg_{x,total}$ denotes the total number of SIs allocated by a_x, t_x^{total} denotes the total number of time steps that a_x has been in existence, while $filled_x(t_x^{total})$ represents the number among those that a_x's capacity was filled with load and some SIs were pending. By multiplying this value with M, it represents the additional load that would have been put on a_x had a_y been a subordinate of a_x since the beginning. The negative sign indicates that this value represents an addition to load, and thereby, a decrease in utility. In a similar fashion, the second term, $\Delta load_y$, is calculated by estimating the possible increase in the load on a_y had a_x been its superior since the beginning. For the third term, $\Delta load_{OA}$, the estimation is carried out by assuming that, had a_y been a subordinate of a_x, all those allocations that started at a_x and ended at a_y via a delegation chain involving other agents, would have been allocated directly. Therefore, the load on the intermediary agents would have been less (this value can be calculated by a_x as it gets back information about the delegation chain of each of the SIs allocated by it). Similarly, the fourth term, $\Delta cost_{comm}$, is also estimated, while the last term, the reorganization cost ($-D$) is a known constant (the minus sign again signifies a decrease in utility). A detailed explanation of the utility calculation, along with the corresponding equations, is available in Section 4.1.1 in Kota [2009]. In this way, using our mechanism, every pair of agents can jointly evaluate the utility for taking any of the possible actions towards changing their relation, at any time step. Being cooperative, the agents do not have conflicts as the value corresponds to the utility of the relation to the organization and not to the individual agents. In a similar context, Sims et al. [2003]

show that adaptation among cooperative agents which use this kind of social utility through sharing of values performs better than self-utility-based methods. Thus, this continuous adaptation of the relations helps in the better allocation of SIs amongst the agents as they will maintain relations with only those agents with whom they require frequent interactions. It is evident that the adaptation method does not need the information about the tasks, but only the resultant agent interactions. Therefore, the method is independent of the task model and allocation mechanisms. Hence, our adaptation method will work for a more constrained or relaxed task model, as long as the information about their personal agent interactions is available to the respective agents.

4.1. Metareasoning

Now, we focus on how an agent can decide which acquaintances to choose for initiating the preceding detailed adaptation process. In the ideal scenario, at line 1 in Algorithm 1, all acquaintances of an agent will be chosen for reasoning about adaptation. However, as the computation required for these utility calculations and reasoning depends on R (Section 3.3), it need not be negligible and might exhaust the computational capacities of the agent that, otherwise, would have been spent on task-related actions (allocation and execution). Thus, when R cannot be ignored, an agent will have to smartly select the acquaintances for *Chosen* in line 1. Thus, effective metareasoning emerges as an important aspect of the adaptation process.

In our case, this problem boils down to the following: at any given time step, an agent should decide on how many and which agents to select for initiating reorganization procedures. This can be viewed as a form of the well-known *coupon collector's problem* [Motwani and Raghavan 1995] and, therefore, we explore a simple randomized approach that is typically used for such problems. In the coupon collector's problem, there are n types of coupons and an infinite number of coupons for each type. At each trial, a coupon is chosen at random. We can map this problem to our scenario by considering every agent to be the collector, and all its acquaintances (including the peers, superiors, and subordinates) as the coupons. Also, in our case, there can be several trials in a single time step.

Now, if X is the number of trials such that at least one coupon of each type is collected, then the expectation of X is: $E(X) = nln(n) + O(n)$ [Motwani and Raghavan 1995]. This assures us that even when chosen randomly, on average, all acquaintances of an agent will be picked up for reorganization deliberation in a given period of time (for 20 acquaintances, this translates to approximately 80 trials). Therefore, an agent can just randomly choose k acquaintances at a time step (in line 1). Moreover, this k can be varied according to the situation. When an agent has free capacity that will otherwise be wasted, k should be such that the whole of the remaining capacity is utilized for reorganization. However, even when the agent is overloaded, reorganization might be necessary. On such occasions, k is based on the percentage of successful reorganizations in the previous time step. In more detail, at a time t, k is determined as

$$k_t = max \begin{cases} 1 & \text{minimum limit} \\ (L_x - l_x)/R & \text{based on free-capacity} \\ acqts_x * changed_{x,t-1}/k_{t-1} & \text{success ratio in prev. iteration} \end{cases} \quad (7)$$

where $acqts_x$ represents the number of acquaintances of a_x, $changed_{x,t-1}$ denotes the number of relations of a_x changed in the previous time step, and k_{t-1} is the k value used in the previous time step. The minimum possible value of k is limited to 1, so that at least one acquaintance is considered for reorganization in any time step. In this way, free capacity is never wasted and, at the same time, an agent will carry out

ALGORITHM 2: In terms of agent a_x applying WoLF

```
1 if W_{x_P} ≠ ∅ then //  Set of pending SIs of a_x
2 |  s ← arg-MaxOccuring{W_{x_P}} AND ∉ AccmSet_x;
3 |  A_s ← agents providing service s;
4 |  a_y ← randomly chosen from A_s;
5 |  form_subr_{x,y};
  end
```

reorganization even when it has a huge number of pending SIs by adapting k according to its need for reorganization.

As described in Section 3.2, organizations can be open and/or dynamic in nature. However, our method, detailed until now, might not work well for such organizations. This is because, in open and dynamic organizations, some of the assumptions of the method will be invalidated. For example, a new agent entering the system will not have any past interactions with the existing agents, thereby resulting in zero value for some of the attributes of the value function. Similarly, when the service set of an agent changes, all of its past interactions will not provide the best picture for its usefulness in the future to other agents. Therefore, now we extend our adaptation method so that it performs well even in such open and dynamic organizations.

4.2. Open Organizations

When a new agent joins the organization, it needs to be assimilated into the structure by the existing ones. However, for an agent to form a relation with a new agent, it has to be able to predict how useful that new agent will be and in what type of relation. This is not straightforward as there are no past interactions with the new agent on which to base any utility calculations (as required by our method). Therefore, the agent is faced with an explore versus exploit trade-off: whether to *explore* by forming an authority or a peer relation with a new agent, or to reorganize with the past agents only by *exploiting* the known information about them. This choice could be tackled by employing specially designed "explorer agents," whose sole task is to monitor the performance of all the agents (including the new ones)[Maximilien and Singh 2005]. However, we do not use such an approach because it requires special agents and that contradicts our self-organization principles. Thus, our intention is to imbibe the adaptation method into the task-solving agents without needing any external help.

Against this background, we find that a context-based exploration strategy is well suited to our problem. The agent looks at its context to decide whether to explore or not. For determining this context, we use the same intuition that is behind the well-known "Win or Learn Fast" [Bowling and Veloso 2001] strategy. The WoLF principle is: "learn quickly while losing, slowly while winning." We use the same basis for differentiating the context at the agents: an agent can either be considered winning (if it has unused capacity) or losing (when it has a pending queue of SIs). Therefore, an agent that is not overloaded will only follow Algorithm 1 by ignoring the new agents joining the system. However, an agent with pending SIs will actively seek new subordinates to be able to improve its delegation of SIs. This addition to the fundamental method is presented as a pseudocode in Algorithm 2.

In more detail, an agent, when overloaded (checking in line 1), identifies which particular service occurs the most in its waiting list that is not supplied by any of its current subordinates (line 2). Then, in line 3, it searches through all of its acquaintances (including the newly entered agents) for those offering that particular service. Finally, in lines 4–5, it forms a superior-subordinate relation with one such randomly chosen

agent.[6] As a result, new agents will be assimilated quickly by the existing agents into the structure. Moreover, these new agents will end up forming the relations where they are most needed, thereby leading to a more equitable distribution of load across the organization. In addition, a new agent offering services which are not much in demand will be ignored (as the agents offering those services are winning anyway) and thus not add any unnecessary management load. In contrast, when an agent leaves, the others can easily reorganize using the method in Algorithm 1 without needing any such additional step.

This approach is useful because the agent is aware of its current context (how quickly it is able to perform the SIs it is receiving) and drawbacks (which services it needs help with) and therefore the exploration is not random but directed (seeking only those acquaintances that can provide these overloaded services). The reverse, in which nonoverloaded agents seek to form new relations with incoming agents, is not productive because the relations at these agents are already sufficient for them to allocate and execute SIs efficiently. Thus adding more relations will only increase their management load and delay tasks with unnecessary delegation without providing any additional benefits. On the same lines, it is also not required for the agents to seek help with services that are already being provided by their existing subordinates. This is because the agent can delegate the SIs to these subordinates, without overloading itself, and it is for the subordinates to seek help if required. In these scenarios, if the subordinate is further delegating the SIs to its own subordinates, then the fundamental method itself, by design, enables the agents to recognize this and adapt accordingly (thereby, reducing the chains of delegation). In summary, for open organizations, the adaptation method includes this context-based exploration strategy, inspired by the "WoLF" principle, in addition to the fundamental method.

4.3. Dynamic Organizations

As explained in Section 3.2, the dynamism of the organizations is caused by the changing service sets of agents. As agents gain new services and/or lose old ones, the relations should be changed accordingly to reflect the changed circumstances. However, our adaptation method is based on all the past interactions between the agents. The method of using the whole history of interactions as guidance for calculating utilities during adaptation might not be the most suitable approach when the agents' service sets are changing, because the kind of interactions that they require might also change.

An obvious approach seems to be to partition utilities on the basis of the services, by grouping the interactions by their services. But such an approach doesn't work because an agent losing a service doesn't exclude it from being able to provide the service via its subordinates and so on (this approach and its fallibility is explained in detail in Section 4.2.2 in Kota [2009]). In contrast, the extension presented earlier (Section 4.2) is somewhat useful for dynamic organizations as well, particularly when agents are gaining new services. In more detail, when some agents gain new services, they can be treated as "new agents" with respect to those services and thus considered for forming subordinate relations by the "losing" or overloaded agents, just as described earlier in Algorithm 2. However, the method is not helpful when agents might be losing services. Moreover, even for the former case of agents gaining services, just using this method will not be sufficient. This is because the agents are already burdened with the whole history of interactions when their service sets were different. Therefore, they might

[6] a_y is chosen randomly as otherwise the decision process will involve interacting and exchanging utilities with all qualifying acquaintances, thus using up more computational capacity at a_x while it is already overloaded.

Table I. Mapping of the Organization Type to Algorithm Required

Org. Type	Closed	Open
Static	fundamental method	fundamental method + exploration (WoLF inspired)
Dynamic	fundamental method with decaying weights + exploration (WoLF inspired)	

not be able to form the best relations for the changed circumstances despite actively seeking specific-service-providing relations using the method.

A more successful approach is to give weights to the past agent interactions depending on the time elapsed since they actually took place. This makes the adaptation more responsive to changing scenarios. By recent interactions contributing more to the utility function than the older ones, the adaptation will reflect the latest scenario rather than the summary of the whole scenario until then. Thus, it will be suitable for dynamic environments in which the kind of required interactions will be changing with time.

In more detail, in the fundamental method, the values for the terms in Eq. (5) are obtained by summing up the relevant agent interactions. During this summation, all interactions were given the same importance. However, in this extended method, weights are assigned to the individual interactions on basis of how far in the past they had occurred. That is, the most recent interaction will have the maximum weight and the older interactions will have lesser weights correspondingly. This decrease in weights will be given by a "decay function." The rate of decay can be tuned according to the rate of dynamism of the organization. A highly dynamic organization (where agents are losing and gaining services at a fast rate) should have a steeper decay function than an organization with a slower rate of change. The pseudocode for this decay mechanism is given in the Electronic Appendix.

The advantage of this approach of using the decay function is that it forces the agents to learn at a faster rate, thus reflecting the increased dynamism of these kinds of scenarios. At the same time, it does not make any additional assumptions and is not specifically dependent on the type of changes to the organization (like changing services or capacities). In this context, it is clear that this decaying weights approach is on the same lines as a fixed rate reinforcement learning method. However, a principled Q-learning approach was found wanting when we experimented with it. Specifically, being based on some reward mechanism, it is unable to recognize the explicit connection between performance and agent interactions. Furthermore, as this setting puts forth an extremely large-scale multi-agent RL problem (a learner is required for each possible pairing of agents), Q-learning is not able to learn useful action-state values despite trying various learning rates and other parameters.

In this section, we have first presented the basics of our adaptation method suitable for closed static organizations. Then, we discussed the extension and modifications required for the method to function well in open and dynamic organizations. This section is succinctly summarized by Table I.

5. EXPERIMENTAL EVALUATION

We demonstrate the effectiveness of our self-organization-based adaptation method through experimental evaluation. We first describe the setup used for the experiments and then present the obtained results. First we discuss the results of the experiments on static closed organizations (Section 5.2) and then for static open organizations (Section 5.3). Later on, we move onto dynamic organizations (Section 5.4). Additional results on the structures formed by adaptation and performance of dynamic open organizations are listed in the Electronic Appendix.

5.1. Experimental Setup

We use the organization model described in Section 3 as our simulation platform. However, we make the assumption that all agents are at least acquainted with each other by default. Therefore, no two agents will be strangers. We do this so to be able to focus solely on evaluating the structural adaptation method and isolate it from being affected by the *service discovery* aspects. For ease of reference, we refer to our fundamental method (without the subsequent enhancements) as `k-Adapt`. Its extension using context-based exploration, inspired by the WoLF principle, as discussed in Section 4.2, is called `wolf-k-Adapt`. Similarly, the modified method presented in Section 4.3 is referred to as `decay-adapt`. Finally, the method incorporating both the exploration strategy and the decay mechanism is referred to as `wolf-decay-Adapt`. To determine the effectiveness of our approach, we compare its performance with two intuitive methods: `Central` and `Random`, that act as the benchmarks. Moreover, to compare with the current state-of-the-art, we modify the token-based agent network adaptation method presented by Glinton et al. [2008] so that it is suitable for agent organizations and use it for comparison as well. As mentioned earlier in Section 2, this is the latest and most relevant work dealing with adaptation in agent networks or organizations in a decentralized fashion. Similarly, we compare with a few other variations of `k-Adapt` to show the importance of all the components of our algorithm. All of these methods are described next.

Central. This is a centralized allocation mechanism containing a central repository that maintains information about the service sets and loads of all the agents in the organization, and is accessible without cost to any agent. The agents do not need to maintain any relations; whenever an agent needs to allocate an SI, it looks up the repository seeking the most suitable agent (capable of the service and having maximum free capacity at the time) and allocates to it. Thus, all allocations are one-step direct delegations, and the agents do not use up any capacity for allocation. This method gives an upper bound on the performance of an organization, but is not a practical or robust solution because it involves maintaining an up-to-date and exhaustive central repository with costless and instantaneous access to all agents.

Random. In terms of the k-Adapt method (Algorithm 1), this strategy involves an agent randomly choosing some of its acquaintances for adaptation (line 1), and then randomly choosing a reorganization action (line 8). Nevertheless, the rate of change is adjusted so that the amount of reorganization is roughly equal to that caused by our method (so that the performance of `Random` is not affected by the aggregation of reorganization cost). Moreover, the number of relations in the organization is maintained at a moderate level by varying the probability of forming or dissolving a relation. The probability to form a relation by an agent is set as being inversely proportional to the existing number of relations of that kind at the agent. Hence, an agent with very few subordinates has a higher chance of forming an additional superior-subordinate relation than an agent with more subordinates and vice versa. In the same way, an agent with more peers has higher probability of dissolving a peer relation than an agent with less peers and so on. This enables the organization to be always reasonably connected. Thus, this method represents a random structural adaptation strategy which does not involve any reasoning (and therefore $R = 0$ for it) and constitutes the lower bound.

Token. Glinton et al. [2008] use a token-based algorithm for decentralized adaptation of links in a task-solving agent network. However, as we discussed in Section 2, their method is not directly applicable to our domain since it only enables the agents to form or delete links, but does not let them to choose between different types of links (or relation-types, as in an organization). Therefore, to fit it into our problem domain, we created two versions of their method: In **Token-all**, the agents use the same set of tokens, irrespective of the relation-types, for adaptation. At the time of forming/dissolving

a relation, they randomly choose the type of relation to modify, whereas in **Token-type**, the agents use a different set of tokens for each relation type. Therefore, the adaptation process for each of the relation-types runs independently of the others, though making sure that there are no conflicts. We use both these versions of their approach for comparison. The token-based algorithm also contains several parameters like TTL (Time-To-Live of the token) and MAX_DEGREE (maximum links allowed at an agent) which we fine-tuned after extensive analyses to obtain the best possible performance and present those results.

free-Adapt. For this method, the reorganization load coefficient R is set to 0. It represents the case when the reorganization can be considered resource-free. Thus, it is the same as `k-Adapt` but differs only in line 1, where all the acquaintances are chosen for reorganization instead of just k. This makes it a theoretical upper bound for the performance of `k-Adapt`.

all-Adapt. This is the same as `free-Adapt`, differing only in R, which is set to the same value as in `k-Adapt` and not 0. Now, while the profit obtained by the organization ($profit_{ORG}$ in Eq. (4)) represents the organization's performance, there are two independent simulation variables that are of interest: (i) distribution of services across agents and (ii) similarity across tasks. Next we discuss them.

Distribution of services across agents. The degree of heterogeneity of the agents in the organization depends on the distribution of services across them. This is a relevant parameter because the significance of the organization structure is greater when the agents are heterogeneous. In such a case, an efficient structure will need to connect every agent with all those agents providing services relevant to it and alongside help in load distribution. In contrast, for homogeneous agents, load distribution is the only feature that can be influenced by the structure. We distributed the services among the agents using a parameter called Service Probability (SP). That is, an agent a_x is allocated a service s_i with a probability SP. Thus, when SP is 0, every agent is capable of a unique service only (as every agent should offer at least one service and every service should be offered by at least one agent). When it is 1, every agent is capable of every service. Since, the services are allocated on the basis of a probability, there is always randomness in the way they are allocated to the agents.

In static organizations, the service sets of the agents are unchanging across a simulation run. Hence, in our experiments for static organizations, we vary SP from 0 to 0.5 only (since we verified that beyond 0.5, when the agents are quite homogeneous, the structures did not influence the performance significantly). However, in dynamic organizations, SP will be changing within a simulation run as the agents gain or lose services. Now, agents can gain or lose services gradually or suddenly. Moreover, they might initially lose services and then start gaining them and vice versa. To capture these various scenarios, we vary SP in the following ways in our experiments: (i) SP increases from 0 to 0.25 at a uniform rate and vice versa; (ii) SP increases at a uniform rate from 0 to 0.25 and then decreases back to 0 and vice versa; (iii) SP changes suddenly from 0 to 0.25 midway through the simulation and the other way round. We implement this variation in SP by adding or removing (depending on the case) services from the service sets of the agents at time steps randomly chosen from a uniform distribution (except in case (iii) where it is midway at $t = 2000$), so that the resultant service distribution conforms to the required value. When an agent loses a service, it is forced to reallocate the SIs in its queue requiring that service and waiting for execution to appropriate related agents as it can no longer perform those SIs itself.

Similarity between tasks. The other simulation parameter of importance is the kind of tasks entering the system. The tasks presented to the organization over the period of a simulation run may be completely unrelated to each other or they may have some common SIs and dependency links. This is interesting because, when tasks are similar,

the organization structure should be able to adapt to the recurring task structures, thereby increasing the efficiency of the organization. Moreover, the presence of similarities in the tasks is an existing phenomenon in the real world faced by computing systems. For our experiments, we determine the similarity between the tasks belonging to a simulation run on the basis of what we call *patterns*: stereotypical task components used to represent frequently occurring combinations of SIs and dependency links. Like tasks, patterns are also composed of SIs, but are generally smaller in size. Instead of creating tasks by randomly generating SIs and creating dependency links between them, tasks can be constituted by connecting some patterns by creating dependency links between the SIs belonging to the patterns. In this way, the dependencies between the SIs may follow some frequent orderings (resulting from the dependencies internal to a pattern occurring in several tasks) and some random dependencies (due to the dependencies created between the patterns). Thus, this method of generation enables us to control the similarity between the tasks using the number of patterns (NoP) as the parameter. In our experiments, we consider two scenarios: (i) completely dissimilar tasks ($NoP = \infty$) and (ii) highly similar tasks ($NoP = 5$). In addition to these, the set of patterns being used within a simulation run can also be varied to represent changing characteristics of the task environment. Experiments based on this are detailed in Section 5.2.5 of Kota [2009].

All our experiments comprise 1000 simulation runs for every data point to achieve statistically significant results. All the results are shown with 95% confidence intervals (the errors bars are very close to the marking symbol in the graphs), obtained by multiplying the standard error by 1.96 (z-test). For every simulation, the set of agents and services is first generated and then the services are assigned to the agents on the basis of SP. Next, the set of tasks is generated using NoP. In our experiments, we use a maximum of 25 initial agents in the organization. We have conducted other experiments with bigger numbers of agents (reaching up to 100) and found similar trends as shown here (some sample results can be found in Section *A*.3 of Kota [2009]). Also, static organizations face 1500 tasks over 2000 time steps to constitute one simulation run, while dynamic organizations face 3000 tasks over 4000 time steps for one simulation run. We provided more simulation time for dynamic organizations so that it is sufficient for the changes in the organization (like changing service sets or task pattern sets) to take place. The tasks arrive at an uniform rate, and are assigned to randomly chosen agents in the organization (those then have to initiate the allocation process for the respective tasks). We do not consider any hot-spots for arrival of tasks because those scenarios are captured by the settings containing similar tasks and heterogeneous agents. This is because in these settings, the particular agents providing those recurring services act as hot-spots since they end up being swamped with the SIs that make up the patterns.

Moreover, we set the total number of services ($|S|$) equal to the number of agents $|A|$ (though the distribution will vary according to SP). We set C at 0.25 and M at 0.5 (so that any allocation process will take up at least half a computational unit). Also, we set R at 0.25 and D at 1. The values of these coefficients are explicitly considered by our adaptation method as part of the utility calculations and, therefore, we did not find any interesting trends by varying them over time or over simulations. The maximum size of a pattern is limited to 8 so that, on average, three patterns are required to compose a task (which can have a maximum of 25 SIs). We observed broadly similar patterns with other parameter settings.

Finally, the set of agents A, is kept constant for closed organizations, while for the open ones, a randomly chosen number of temporary agents are added, as described in Section 3. The results shown here are of experiments where the start-times and lifetimes are chosen from a uniform distribution. However, we also conducted experiments

with a combination of distributions for start-times (uniform and normal) and life-times (normal and geometric) and found the resultant trends similar.

We present the results in terms of the percentage of the maximum profit that is obtained by the organization (averaged over the 1000 simulation runs as described before). The maximum profit is given by the profit obtained by `Central` as it represents the theoretical upper bound. For static organizations, the results are presented as graphs plotting the profit obtained for the methods over an increasing SP along the x-axis (increasing the homogeneity of agents). However, for dynamic organizations, SP itself varies within a simulation run. Therefore, the results are presented in a table format for each of the scenarios depicting a particular kind of variance in SP.

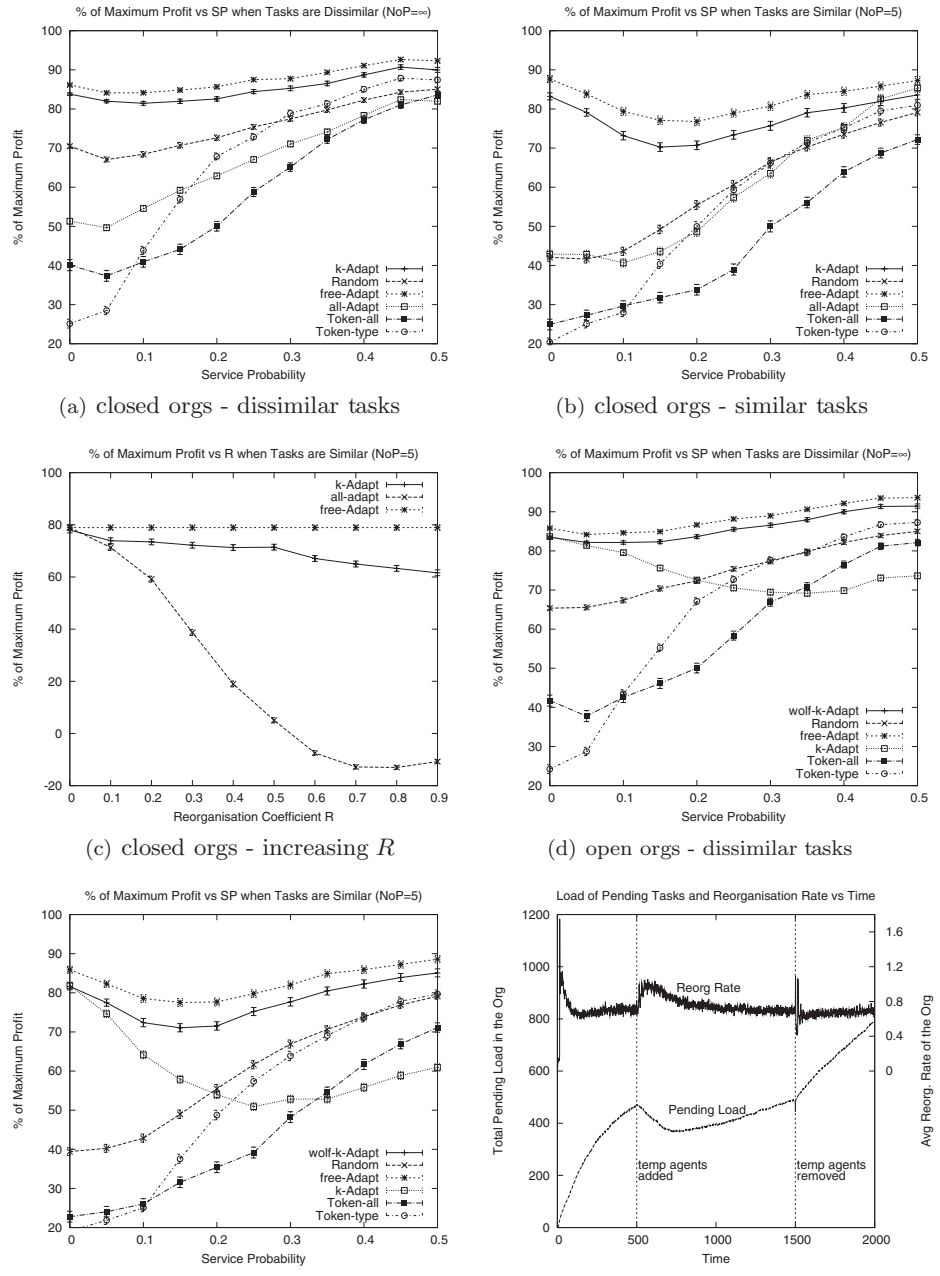
5.2. Results for Static Closed Organizations

Observing the results for static closed organizations, we find that in both the scenarios with dissimilar (Figure 4(a)) and similar tasks (Figure 4(b)), `k-Adapt` performs consistently better than `Random`, `Token-all`, and `Token-type`. The difference in their performance narrows down (from the peak of 40% of profit to 10%) as the similarity of agents increases. This is because a smart method is correspondingly less useful when all the agents are homogeneous, as the significance of the structure itself diminishes. As the agents become homogeneous and everyone can provide most of the services, the task allocation problem reduces to a load distribution problem. In this case, random policies will perform well too because the incoming load (or tasks in our case) is also randomly distributed amongst the agents. Also, we see that `k-Adapt` and `free-Adapt` perform better when $SP = 0$ than for slightly higher values of SP because, as SP increases and more agents are capable of a given service, `Central` continues performing perfect allocations (as it has up-to-date information about loads on all agents), while the agents in the organizations using our method have no way of knowing which relations have free capacities. However, the performance increases for higher values of SP because the average capacity available for any given service becomes larger as agents are capable of more services, thus leading to better task completion times. This is also the reason why `Random`, `Token-all`, and `Token-type` improve with increasing SP. It is also noticeable that `Token-type` improves at a faster rate than `Token-all`. This supports our contention that when different types of relations (or links) are possible between the agents, considering them distinctly during adaptation provides a better performance. We also conducted experiments by varying R from 0 to 0.9 (see Figure 4(c)) and found that the fall in the performance of `k-Adapt` is gradual and minimal, while it is drastic in `all-Adapt`. In fact, for higher values of R, the profit of `all-Adapt` goes below 0, meaning the cost is more than the reward obtained. This shows that metareasoning is a crucial aspect in an adaptation process and cannot be ignored. We see that the performance of `k-Adapt` is always close to that of `free-Adapt`, thus confirming the efficacy of our metareasoning approach.

5.3. Results for Static Open Organizations

In the case of static open organizations, we find that `wolf-k-Adapt` performs considerably better than `Random`, `Token-all`, and `Token-type` when tasks are both dissimilar (Figure 4(d)) and similar (Figure 4(e)). More importantly, `k-Adapt`, which does not make use of exploration based on the WoLF principle, degrades rapidly as the similarity between agents increases. This shows that the context-based exploration is very useful for assimilating new agents into the organization and maintaining the performance.

Furthermore, Figure 4(f) gives us an insight into what is happening to the organization when the agents are added and removed. For this experiment, we fixed the start-time at 500 and life-time at 1000 for the temporary agents. The graph shows the sum of the computations of all pending SIs in the organization (left y-axis) across



(a) closed orgs - dissimilar tasks

(b) closed orgs - similar tasks

(c) closed orgs - increasing R

(d) open orgs - dissimilar tasks

(e) open orgs - similar tasks

(f) open orgs - rending load & reorg. tate

Fig. 4. Results for static organizations.

the time duration of the simulation, and shows the corresponding reorganization rate in terms of the number of relations changed in a time step (right y-axis). For these experiments, we fixed $SP = 0.20$ and $NoP = 0$. We observe a gradual fall in the load starting at time $= 500$ corresponding to when temporary agents are added. Also at

Table II. Profit for Dynamic Closed Organizations with Dissimilar Tasks

SP variance	wolf-decay-Adapt	wolf-k-Adapt	Random	Token-all	Token-type
$0 \rightarrow 0.25$ gradually	93.5% ±0.3	88.6% ±0.3	76.2% ±0.4	46.3% ±1.3	58.6% ±0.9
$0.25 \rightarrow 0$ gradually	88.2% ±0.4	77.9% ±0.4	72.4% ±0.5	42.1% ±1.3	50.2% ±0.9
$0 \rightarrow 0.25 \rightarrow 0$ gradually	92.1% ±0.3	86.5% ±0.3	74.8% ±0.4	44.8% ±1.3	54.7% ±0.9
$0.25 \rightarrow 0 \rightarrow 0.25$ gradually	90.4% ±0.3	80.9% ±0.3	74.6% ±0.4	45.4% ±1.3	57.7% ±0.9
$0 \rightarrow 0.25$ at $t = 2000$	91.8% ±0.3	87.3% ±0.3	74.2% ±0.4	44.1% ±1.3	48.7% ±1.0
$0.25 \rightarrow 0$ at $t = 2000$	90.7% ±0.3	80.4% ±0.4	74.8% ±0.4	51.6% ±1.2	53.5% ±0.8

Table III. Profit for Dynamic Closed Organizations with Similar Tasks

SP variance	wolf-decay-Adapt	wolf-k-Adapt	Random	Token-all	Token-type
$0 \rightarrow 0.25$ gradually	74.7% ±1.1	70.0% ±1.7	55.7% ±1.9	28.6% ±1.4	40.4% ±1.2
$0.25 \rightarrow 0$ gradually	78.8% ±1.0	67.7% ±1.0	47.1% ±0.9	22.6% ±1.3	31.8% ±1.0
$0 \rightarrow 0.25 \rightarrow 0$ gradually	76.5% ±1.0	70.8% ±1.0	53.7% ±1.0	29.1% ±1.3	38.4% ±1.1
$0.25 \rightarrow 0 \rightarrow 0.25$ gradually	76.3% ±0.9	66.3% ±1.0	50.6% ±1.0	26.8% ±1.3	35.1% ±1.1
$0 \rightarrow 0.25$ at $t = 2000$	76.3% ±1.0	71.4% ±1.0	51.3% ±1.1	22.7% ±1.4	27.9% ±1.1
$0.25 \rightarrow 0$ at $t = 2000$	82.1% ±0.9	71.1% ±0.9	50.1% ±1.0	32.7% ±1.2	35.5% ±0.9

time = 1500, there is a quick drop and immediate increase because, when the temporary agents leave, the SIs pending at them are reassigned to the permanent agents. This reassignment requires at least a time step, after which only they are visible as pending load again. Also, the rate of growth of pending load is higher once the agents leave (as seen by the higher gradient). Looking at the reorganization rate, we find that it is high in the beginning and then settles down to an almost uniform rate. Later, there is a sudden jump in the rate when the agents are added and this gradually falls back to the earlier value at around time = 700. This shows that our adaptation process is able to reach its earlier stable state in reasonable time. As expected, we also find another blip in the rate when the agents are removed. This time, it settles much more quickly as the permanent agents are able to easily reform the older structure that existed prior to the addition of the temporary agents.

5.4. Results for Dynamic Closed Organizations

We see that for dynamic closed organizations, `wolf-decay-Adapt` consistently performs significantly better than the other methods in both the scenarios with dissimilar (Table II) and similar (Table III) tasks. It is notable that for dissimilar tasks, `wolf-decay-Adapt` is able to reach 90% of the maximum profit which is 5–10% better than `wolf-k-Adapt` and 15% better than `Random`, while being around 35–40% better than the `Token` methods. In this context, we also observe that the performance of `wolf-k-Adapt` when the service probability SP (introduced in Section 5.1) is reduced is worse than when SP is increased. This is because in organizations using `wolf-k-Adapt`, the agents form relations on the basis of all of their past allocations. However, when the agents start losing services, some of those allocations are no longer possible. However, the agents continue to maintain the relations due to the burden of the long history, thereby reducing their efficiency. This is not the case with increasing SP where agents gain services because allocations that happened in the past will still be possible. Of course, newer kinds of allocations will also be possible which `wolf-decay-Adapt` is capable of identifying much more quickly than `wolf-k-Adapt` by giving more weight to recent interactions.

In these set of results and also in all of the following, we find that the performance of all the methods is better for dissimilar tasks than similar ones. This is because, for similar tasks, the load is high on the particular agents providing those more frequent services and this load cannot be distributed as equitably by the agents with their local views as the `Central` method can with its global view. In a similar vein, we also notice that the gap in the performance between the "adapt" methods and `Random` or "token"

methods is much more for similar than dissimilar tasks. This reinforces our assertion that our adaptation approach is able to identify the patterns across tasks (when they occur) and adapt the structure according to them in an emergent fashion (since the agents are only adapting locally). Another interesting phenomenon to observe is that varying SP gradually over the simulation or suddenly in the middle of the simulation does not affect the performance of any of the methods significantly. We believe this is because the effects of agents gaining/losing services slowly over the total time period averages out to result in the same kind of performance when agents are gaining/losing all of the services only at the middle of the time period.

In summary, we find that, on average, our adaptation method performs at 80% of the omniscient centralized allocation method. Furthermore, on average, it is 20% better than a random reorganization approach (reaching up to a maximum of 45%). More importantly, it is 40% better on average than the token-based methods (reaching a maximum of 60%). The results for open and dynamic organizations show that the respective enhancements, context-based exploration inspired by the WoLF principle and decaying weights, are crucial for maintaining the performance. We see that, on average, the performance of the method with the respective enhancement is 8% better than without it.

6. CONCLUSIONS

This article addresses the problem of developing decentralized structural adaptation methods for problem solving agent organizations based on the paradigm of self-organization. More specifically, using a simple organization model as a framework, we presented a structural adaptation method that can be applied individually and locally by all the agents in order to improve the organization's performance. In agent organizations, the structure is defined by the relations between the agents and a particular relation existing between two agents affects both the involved agents. Therefore, using our method, a pair of agents jointly estimate the utility of changing their inter-relation and take the appropriate action accordingly. Moreover, our method also enables an agent to metareason about when and with whom to initiate this adaptation deliberation. Additionally, we extend our method so that it performs well even in open organizations which have agents moving in and out of the system. The extension enables poorly performing agents to actively seek out suitable relations among the new agents entering the system and then delegate some of their excess load to them. We also modified our method to tackle dynamic organizations wherein the properties of the agents might be changing with time. For such dynamic scenarios, the weights associated with the past interactions of the agents (during utility calculations) decay with time. Therefore, more recent interactions contribute more to the utility than older ones, thus helping the agents keep up with the changes to the agent properties.

We experimentally evaluated our approach by varying interesting parameters like heterogeneity of agents and similarity of tasks, and the openness and dynamism of the organizations. We found that our method performs at 80% (average) of a centralized omniscient method while the current decentralized agent-network adaptation methods only manage to reach 40% (average). Both the enhancements to our basic method, context-based exploration and decaying weights, are found useful for maintaining the good performance in the face of open and dynamic organizations.

It is evident that our adaptation method works purely by redirecting agent interactions, thereby changing the organization structure. A key advantage of this approach is that it does not entail any modifications to the agents themselves or their internal characteristics. Therefore, it is applicable even in situations where the internal properties of the agents (like computation capacity or services provided) in the organizations cannot be altered by the adaptation method. Moreover, the method inherently takes into

account the cost of adaptation and the cost of reasoning about adaptation in addition to the achievable improvement to the organization through adaptation. This makes it suitable for any kind of environment as long as the constants of the environment are known. Since the adaptation method is purely agent-based, decentralized, and continuous over time, it satisfies the principles of self-organization discussed in Section 1. Also, having been developed on an abstract organization platform, the method is generic and applicable to cooperative multiagent systems requiring sustained interagent interactions for achieving task objectives in a resource-constrained environment. It is useful if these interactions are resource-intensive and are regulated by some kind of an organization structure.

The aforementioned characteristics mean that our adaptation method can be used by the individual components of a distributed computing system to manage themselves, as it will enable them to continuously adapt their interactions with the other components in the system in a local and robust fashion. Hence, the work documented in this article demonstrates a simple and robust, decentralized approach for continuous self-adaptation of problem-solving agent organizations, thereby providing an important component for the development of autonomic systems.

Though our organization model was sufficient to serve as a platform for developing the adaptation method, it also provides some avenues for future work. For example, currently the performance is measured by cost and task completion times and does not take into account aspects like quality of service and does not limit other resources like memory and network bandwidth. Extending the model to incorporate these features might throw up more challenges to the adaptation method, like optimizing against different types of constraints concurrently or improving from among the different set of performance measures available.

In a similar vein, our adaptation method makes no assumptions about knowing anything about the dynamism of the system. It functions solely on the history of interactions and does not make use of any information (if available) about the kind of tasks that might be coming in the future or the kind of changes expected in the organization's agents. We plan to extend our method such that the agents can adapt *proactively* when such information is available to them. For example, if they are provided with a probability distribution of the kind of task patterns expected to be seen in the future, it can be taken into account during adaptation so that the organization structure is better prepared to handle these tasks.

By focusing on an abstract model, we have managed to develop a generic adaptation method and tested it empirically in a similarly generic fashion. Nevertheless, the applicability of the adaptation methods also needs to be tested in real-life scenarios. Though autonomic systems are not prevalent as yet, there exist grid systems that perform extensive work-flow-based tasks like large-scale complex scientific calculations or supply-chain and procurement processes for large businesses. The adaptation method could be incorporated in any such suitable distributed computing system and verified whether it helps in improving the performance. Doing this will not only reaffirm the results presented here, but also possibly uncover newer challenges that might crop up during deployment in real-life systems.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

REFERENCES

ABDALLAH, S. AND LESSER, V. 2007. Multiagent reinforcement learning and self-organization in a network of agents. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multi-Agent Systems (Aamas'07)*. 172–179.

ALEXANDER, G., RAJA, A., DURFEE, E. H., AND MUSLINER, D. J. 2007. Design paradigms for meta-control in multi-agent systems. In *Proceedings of the Workshop on Metareasoning in Agent-based Systems at AAMAS'07*. 92–103.

BERNON, C., CHEVRIER, V., HILAIRE, V., AND MARROW, P. 2006. Applications of self-organising multi-agents systems: An initial framework of comparison. *Informatica 30,* 1, 73–82.

BISKUPSKI, B., DOWLING, J., AND SACHA, J. 2007. Properties and mechanisms of self-organizing manet and p2p systems. *ACM Trans. Auton. Adapt. Syst. 2,* 1, 1.

BONGAERTS, L. 1998. Integration of scheduling and control in holonic manufacturing systems. Ph.D. thesis, PMA/K.U. Leuven.

BOU, E., LOPEZ-SANCHEZ, M., AND RODRIGUEZ-AGUILAR, J. A. 2006. Self-Configuration in autonomic electronic institutions. In *Coordination, Organization, Institutions and Norms in Agent Systems Workshop at ECAI'06*. 1–9.

BOWLING, M. AND VELOSO, M. 2001. Rational and convergent learning in stochastic games. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*. 1021–1026.

CAPERA, D., GEORGE, J.-P., GLEIZES, M.-P., AND GLIZE, P. 2003a. The AMAS theory for complex problem solving based on self-organizing cooperative agents. In *Proceedings of the 12th International Workshop on Enabling Technologies (WETICE'03)*. IEEE Computer Society, Los Alamitos, CA, 383.

CAPERA, D., GLEIZES, M. P., AND GLIZE, P. 2003b. Self-organizing agents for mechanical design. In *Engineering Self-Organising Systems*. Lecture Notes in Computer Science, vol. 2977, Springer, 169–185.

CONITZER, V. 2008. Metareasoning as a formal computational problem. In *Proceedings of the Workshop on Metareasoning: Thinking about Thinking at AAAI'08*.

DE WOLF, T. AND HOLVOET, T. 2003. Towards autonomic computing: Agent-Based modelling, dynamical systems analysis, and decentralised control. In *Proceedings of the 1st International Workshop on Autonomic Computing Principles and Architectures*. 10–20.

DELOACH, S. A., OYENAN, W. H., AND MATSON, E. T. 2008. A capabilities-based model for adaptive organizations. *Auton. Agents Multi-Agent Syst. 16,* 1, 13–56.

DI MARZO SERUGENDO, G., GLEIZES, M.-P., AND KARAGEORGOS, A. 2005. Self-Organization in multi-agent systems. *Knowl. Engin. Rev. 20,* 2, 165–189.

DI MARZO SERUGENDO, G., GLEIZES, M.-P., AND KARAGEORGOS, A. 2006. Self-organisation and emergence in multi-agent systems: An overview. *Informatica 30,* 1, 45–54.

DIGNUM, V. 2003. A model for organizational interaction: Based on agents, founded in logic. Ph.D. thesis, Proefschrift Universiteit Utrecht.

FERBER, J. AND GUTKNECHT, O. 1998. A meta-model for the analysis and design of organizations in multi-agent systems. In *Proceedings of the 3rd International Conference on Multi Agent Systems (ICMAS'98)*. IEEE Computer Society, Los Alamitos, CA, 128–135.

FISCHER, K. 2005. Self-organisation in holonic multiagent systems. In *Mechanizing Mathematical Reasoning*, vol. 2605, Springer, 543–563.

FORESTIERO, A., MASTROIANNI, C., AND SPEZZANO, G. 2008. A self-organizing grid featuring bio-inspired algorithms. *ACM Trans. Auton. Adapt. Syst. 3,* 2, 1–37.

GASTON, M. E. AND DESJARDINS, M. 2005. Agent-Organized networks for dynamic team formation. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05)*. ACM, New York. 230–237.

GERSHENSON, C. 2007. Design and control of self-organizing systems. Ph.D. thesis, Vrije Universiteit Brussels.

GLINTON, R., SYCARA, K. P., AND SCERRI, P. 2008. Agent organized networks redux. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*. AAAI Press, 83–88.

HANNOUN, M., BOISSIER, O., SICHMAN, J. S., AND SAYETTAT, C. 2000. Moise: An organizational model for multi-agent systems. In *Proceedings of the International Joint Conference, 7th Ibero-American Conference on AI, 15th Brazilian Symposium on AI (IBERAMIA/SBIA '2000)*. Lecture Notes in Artifical Intelligence, vol. 1952, Springer, 152–161.

HILAIRE, V., KOUKAM, A., AND RODRIGUEZ, S. 2008. An adaptative agent architecture for holonic multi-agent systems. *ACM Trans. Auton. Adapt. Syst. 3,* 1, 1–24.

HOGG, L. M. AND JENNINGS, N. R. 2001. Socially intelligent reasoning for autonomous agents. *IEEE Trans. Syst. Man Cybernet. 31,* 5, 381–393.

HOOGENDOORN, M. 2007. Adaptation of organizational models for multi-agent systems based on max flow networks. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*. AAAI Press, 1321–1326.

HORLING, B., BENYO, B., AND LESSER, V. 2001. Using self-diagnosis to adapt organizational structures. In *Proceedings of the 5th International Conference on Autonomous Agents (AGENTS'01)*. ACM Press, New York. 529–536.

HUBNER, J. F., SICHMAN, J. S., AND BOISSIER, O. 2004. Using the MOISE+ for a cooperative framework of MAS reorganisation. In *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence (SBIA'04)*. vol. 3171. Springer, 506–515.

ISHIDA, T., GASSER, L., AND YOKOO, M. 1992. Organization self-design of distributed production systems. *IEEE Trans. Knowl. Data Engin. 4,* 2, 123–134.

JACKSON, M. O. AND WATTS, A. 2002. The evolution of social and economic networks. *J. Econ. Theory 106,* 2, 265–295.

JIN, Y. AND LEVITT, R. E. 1996. The virtual design team: A computational model of project organizations. *Comput. Math. Organ. Theory 2*, 171–196(26).

KAMBOJ, S. AND DECKER, K. S. 2007. Organizational self-design in semi-dynamic environments. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'07)*. 1220–1227.

KEPHART, J. O. AND CHESS, D. M. 2003. The vision of autonomic computing. *IEEE Comput. 36,* 1, 41–50.

KOTA, R. 2009. Self-Adapting agent organisations. http://eprints.soton.ac.uk/72019/.

KOTA, R., GIBBINS, N., AND JENNINGS, N. R. 2008. Decentralised structural adaptation in agent organisations. In *Proceedings of the Workshop on Organised Adaptation in Multi-Agent Systems at AAMAS '08*. 1–16.

MANO, J.-P., BOURJOT, C., LOPARDO, G., AND GLIZE, P. 2006. Bio-Inspired mechanisms for artificial self-organised systems. *Informatica 30,* 1, 55–62.

MATHIEU, P., ROUTIER, J.-C., AND SECQ, Y. 2002. Principles for dynamic multi-agent organizations. In *Proceedings of the 5th Pacific Rim International Workshop on Multi Agents*. Springer, 109–122.

MAXIMILIEN, E. M. AND SINGH, M. P. 2005. Multiagent system for dynamic Web services selection. In *Proceedings of 1st Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE) at AAMAS'05*. 25–29.

MOTWANI, R. AND RAGHAVAN, P. 1995. *Randomized Algorithms*. Cambridge University Press.

NORMAN, T. J., PREECE, A., CHALMERS, S., JENNINGS, N. R., LUCK, M., DANG, V. D., NGUYEN, T. D., DEORA, V., SHAO, J., GRAY, A., AND FIDDIAN, N. 2004. Agent-Based formation of virtual organisations. *Int. J. Knowl. Based Syst. 17,* 2-4, 103–111.

RAJA, A. AND LESSER, V. 2004. Meta-level reasoning in deliberative agents. In *Proceedings of the Intelligent Agent Technology (IAT'04), IEEE/WIC/ACM International Conference*. IEEE Computer Society, Los Alamitos, CA, 141–147.

SCHLEGEL, T. AND KOWALCZYK, R. 2007. Towards self-organising agent-based resource allocation in a multi-server environment. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'07)*. ACM, 1–8.

SIERRA, C., RODRIGUEZ-AGUILAR, J. A., NORIEGA, P., ESTEVA, M., AND ARCOS, J. L. 2004. Engineering multi-agent systems as electronic institutions. *Euro. J. Inf. Profess. V,* 4, 33–39.

SIMS, M., GOLDMAN, C., AND LESSER, V. 2003. Self-Organization through bottom-up coalition formation. In *Proceedings of 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS'03)*. ACM Press, 867–874.

TESAURO, G., CHESS, D. M., WALSH, W. E., DAS, R., SEGAL, A., WHALLEY, I., KEPHART, J. O., AND WHITE, S. R. 2004. A multi-agent systems approach to autonomic computing. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*. IEEE Computer Society, Los Alamitos, CA, 464–471.

VAZQUEZ-SALCEDA, J., DIGNUM, V., AND DIGNUM, F. 2005. Organizing multiagent systems. *Auton. Agents Multi-Agent Syst. 11,* 3, 307–360.

WANG, Z. AND LIANG, X. 2006. A graph based simulation of reorganization in multi-agent systems. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'06)*. IEEE Computer Society, Los Alamitos, CA, 129–132.

WATTS, A. 2001. A dynamic model of network formation. *Games Econ. Behav. 34,* 2, 331–341.

Received June 2009; revised April 2010; accepted August 2010