UNIVERSITY OF SOUTHAMPTON
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

Self-Adapting Agent Organisations

by Ramachandra Kota

Supervisors: Prof. Nicholas R. Jennings and Dr. Nicholas Gibbins
Examiners: Prof. Mark d'Inverno and Dr. Enrico H. Gerding

A thesis submitted in partial fulfillment for the degree of
Doctor of Philosophy

November 2009

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Ramachandra Kota

Autonomic systems, capable of self-management, are being advocated as a solution to the problem of maintaining modern, large, complex computing systems. Given this, we believe self-organising multi-agent systems provide a convenient paradigm to develop these autonomic systems because such self-organising systems can arrange and re-arrange their structure autonomously, without any external control, in order to adapt to changing requirements and environmental conditions. Furthermore, such systems need to be decentralised, so that they are robust against failures; again, this characteristic fits with the multi-agent paradigm. With this motivation, this thesis explores the area of self-organisation in agent systems, and particularly focuses on the decentralised structural adaptation of agent organisations.

In more detail, self-organisation has been generated in agent systems using various approaches like stigmergy, reinforcement mechanisms, cooperative actions of agents and reward based mechanisms for selfish agents. However, none of these are directly applicable to agent organisations because they cannot be incorporated into deliberative agents working towards organisational goals. The few adaptation mechanisms that are applicable are either centralised or are based on restricted settings and also ignore the resources being used by the adaptation process. Here, we particularly focus on such problem solving agent organisations because they provide a suitable representation for autonomic systems. We investigate and develop mechanisms to incorporate decentralised structural adaptation in organisations to improve their performance.

More specifically still, we provide a generic framework for representing problem solving agent organisations. This serves as the platform on which we investigate approaches for structural adaptation. Following this, we demonstrate a robust, decentralised adaptation method that enables the agents to modify the organisational structure. As the method is based on self-organisation principles, the agents use only their local views to change their structural relations to achieve a better allocation of tasks in the organisation. Particularly, 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 organisations using our method is close ($70-90\%$) to that of an idealised centralised allocation method and is considerably better ($10-45\%$) than the current state of the art decentralised approaches.

Contents

List of Figures

List of Tables

List of Algorithms

Declaraction of Authorship

I, *Ramachandra Kota* declare that the thesis entitled *Self-Adapting Agent Organisations* and the work presented in it are my own. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published in a number of publications (see Section 1.4 for a list).

Signed:

Date:

Acknowledgements

Research is not generally possible in solitude. As any other person, I have been greatly helped in the course of my PhD by several people and I am thankful to all of them. Foremost, I would like to express my heartfelt gratitude to my supervisors Prof. Nick Jennings and Dr. Nick Gibbins for their immense contribution towards this thesis. Their constant support, peppered with insightful advice and suggestions, not only refined my research direction but also enhanced my outlook and the way of perceiving the various aspects involved in any topic or issue. Their inputs and comments (often in the form of "why?'s" and "justify's") on the versions of my papers and reports have sharpened my writing and made it more lucid and precise. Frankly, this thesis would not have been possible without their guidance and supervision.

I would also like to acknowledge Dr. Enrico Gerding for his invaluable remarks as the examiner for both my nine-month report and the MPhil to PhD transfer report. I also owe thanks to my research group within the IAM lab which is filled with outstanding individuals possessing both a great research acumen and an extremely friendly and helpful nature. Particularly, working with Archie Chapman has been a very good learning experience and also acted as a productive diversion from my research topic. He has also helped me on several occasions on a variety of issues ranging from latex to administrative procedures. I am also thankful to my other two "PhD batch-mates" and "bay-mates" of 3 years, Simon Williamson and Athanasios Papakonstantinou for making my work environment more fun and enjoyable. Dr. Sebastian Stein and Dr. Perukrishnen Vytelingum have always been readily helpful whenever I approached them with any doubts or questions.

My PhD experience was not just about the research but also about living in the UK. My time here was made comfortable and pleasant by the wide circle of friends I luckily managed to possess. You all know who you are, and I thank each one of you for the camaraderie. Working on a computer, most of the time, can also get quite lonely but that never happened because of all my friends, who despite being spread over different parts of the world, were ever present online. The daily chats with them provided not just the much needed respite but also acted as a constant source of motivation to work.

Finally, I cannot sufficiently thank my parents and brother for all the love that they have showered upon me throughout my life. Without their unflinching faith, abundant care and the encouragement to pursue whatever I chose, writing a PhD thesis would not have even been a dream, let alone a reality!

Chapter 1

Introduction

As computing systems get ever larger and more complex, they are becoming increasingly interconnected and correspondingly more difficult to maintain. Due to the increase in the size, complexity and the number of components, it is no longer practical to anticipate and model all possible interactions and conditions that the system may experience at design time. Similarly, the systems are becoming too large and too complex for system managers to maintain them at run-time. To tackle these problems, several researchers have argued that such large complex systems should be *autonomic* — that is, the computing systems should manage themselves (Kephart and Chess, 2003; Mainsah, 2002). Specifically, autonomic systems are expected to maintain and adjust their operations according to changing requirements, demands, resources, other external conditions and failures. In short, autonomic systems possess the capability of self-management. Moreover, this self-management behaviour of autonomic systems should arise in a decentralised manner, through the interactions between its individual components along with the internal self-management properties of the components. By so doing, the autonomic system will be more robust as there won't be a single point of failure. Such decentralised autonomic systems exhibiting self-management properties, by definition, have to be autonomous and proactive. Therefore, a multi-agent systems approach is well suited for developing autonomic computing systems (Tesauro et al., 2004) as agents are also autonomous and proactive by nature. Multi-agent systems also provide a suitable paradigm for decentralised systems in which autonomous individuals engage in flexible high-level interactions. Thus, the self-management principles of autonomic systems can be mapped onto the notion of agents by considering the components of the system to be autonomous agents engaging in interactions to produce an autonomous self-managed system (De Wolf and Holvoet, 2003). This naturally gives rise to the need for developing agents that are capable of self-management both individually and collectively. Thus, it becomes necessary to explore multi-agent systems that can exhibit this self-management.

Now, the area of research that deals with self-management in agent systems is *self-organisation*, *the mechanism or the process enabling the system to change its organisation without explicit external command during its execution time* (Di Marzo Serugendo et al., 2005*a*). Building on this, we contend that self-organising multi-agent systems provide a suitable paradigm to develop

autonomic systems. Specifically, such self-organising systems can arrange and re-arrange themselves autonomously, without any external control, in order to adapt to changing requirements and environmental conditions, similar to the self-management expected in autonomic systems. Therefore, to further aid the development of autonomic systems, our work explores the area of self-organisation in systems of autonomous agents. In more detail, any self-organising system is expected to have the following properties (Di Marzo Serugendo et al., 2005*b*):

- No external control: all of the adaptation process is initiated internally, that is, only by the components of the system and only changes the internal state of the system.
- Dynamic operation: the system is expected to evolve with time; self-organisation is a continuous process.
- No central control: the organisation is maintained only through local interactions of the individual components with no central guidance.

Given that the self-management properties ought to arise in the system through the interactions between the individual components, it is required that the components of the system are allowed the freedom to adapt their interactions with the other components. In particular, adapting these interactions is necessary because, purely changing the internal characteristics of the components will not be sufficient for improving performance as most of the tasks and goals facing a distributed system involve multiple components and interactions across them. For example, consider the interconnected network of a university as a form of an autonomic grid system. Being a university, it contains various labs with their own specialised computing systems, as part of the overlaying network of the university. That is, there might be a graphics lab containing computers with some high end graphics cards and drivers for rendering rich and intensive images or videos. Similarly, some computers in the geography lab might contain various GIS maps and specialised software for their interpretation. In the same way, there would be such distinct service providing computers in the other departments like sociology, mathematics, physics and so on. In this context, there will be complex computing tasks that need several computers (possibly situated in different labs) providing specialised services for their accomplishment. A task might need statistical analysers from the mathematics department for analysing data available from the sociology department in order to predict natural resource, like water and wood, usage as needed by the institute on environmental conservation. Thus, the computers on the university network, providing different services, need to interact with one another to perform these complex tasks. 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, optimise the overall performance.

Having established the usefulness and necessity of developing self-organisation for distributed computing systems, in the next section, we show that representing such systems as a problem-solving agent organisation will aid in the development of the required self-organisation techniques. Following that, in Section 1.2, we argue that decentralised structural adaptation is an appropriate approach for incorporating self-organisation characteristics into such organisations. With that, we motivate the purpose of this thesis and state the research objectives in Section 1.3. Following that, we discuss our research contributions in Section 1.4 and conclude this chapter with a description of the structure of the thesis in Section 1.5.

1.1 Problem-solving Agent Organisations

In the context of distributed systems, the social interactions of the components can be quite ad-hoc and not guided by definite regulations or they can be structured using an explicitly depicted network or organisation. In the latter case, the individual components of the system will be modelled as autonomous agents participating in an organisation and the interactions between the components are governed by the structure of this organisation. We focus on this approach because, regulating the interactions in the system through the organisation structure will aid in the design of adaptation techniques by suitably representing the recurring interactions between the components. For example, consider the autonomic system being used to maintain the computing systems in a university, as discussed earlier. Now, given the large number of computers or components in the system, one computer can hope to maintain links with just a limited number of those present in the network. Therefore, say a computer in the geography lab regularly needs computers equipped with good graphics capabilities for rendering its maps. It has to choose between maintaining links with just one computer or with many in the graphics lab. The former case will lead to less processing at the geography computer because it doesn't have to choose which graphics computer to allocate its tasks, but at the same time, might lead to delays when that particular graphics computer is busy or reallocates the task to another computer. In contrast, the latter case of maintaining links with all graphics computers will require more processing at the geography computer every time it has to allocate a task, as it has to specifically evaluate the capability of each linked computer before deciding the allocation. At the same time, it might help in getting quicker outputs once the task is allocated well to a suitable computer. Now, if provided with the structure, the geography computer can smartly choose how many graphics computers to maintain links with, by evaluating the possible delays that might occur when accessing most of the graphics computers indirectly and compare that with the resources saved at itself in terms of processing cycles per each allocation task.

Once the social interactions are explicitly depicted by the organisation structure, any approach seeking to embed adaptation into the system can then use and focus on this organisation as a whole rather than working on each of the individual components separately. Thus, the organisation model will provide a better overview of the global performance of the system without

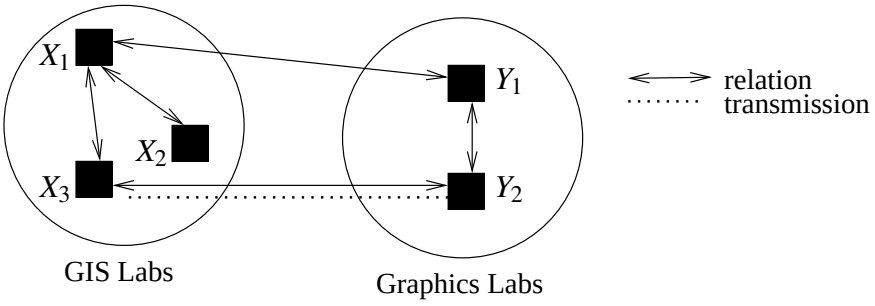
compromising on the individuality of the constituent entities. That is, when the example system presented above is provided with an organisation model, the individual computers in the different labs need not be meddled with for bringing in autonomic properties into the system. Rather, only the overlying middleware, used to interact with the network, that is common to all the computers, needs to be worked on. On this basis, we argue that a formally modelled organisational representation of the components will help in managing their social interactions (Zambonelli et al., 2003), and, at the same time, provide insights into possible avenues for self-organisation and adaptation towards improving global behaviour. More specifically, we contend that depicting the distributed computing systems, including the service providers, their social interactions and the task environment, using an abstract organisation framework will provide a suitable platform to develop and evaluate self-organisation and other techniques attempting to bring about autonomic characteristics like self-management into the system.

Against this background, we are primarily interested in multi-agent systems that act as cooperative problem-solving organisations (i.e. those comprising cooperative autonomous agents that receive inputs, perform tasks and return results). The problem-solving part of this definition is in contrast to organisations that just provide guidelines to be obeyed by agents participating in the organisation to achieve their individual goals (an example of such an organisation model is provided by Sierra et al. (2004)). Specifically, these organisations 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 are not suitable to be mapped onto distributed computing systems. The cooperative part of the definition is in contrast to those comprising self-interested and often competing agents (Nisan et al., 2007). We chose to use such cooperative problem-solving agent organisations because they can be decentralised with autonomous and independent agents which accomplish tasks by providing services and collaborating with each other through social interactions governed by the organisation's structure. Thus, it models the salient features of distributed computing systems and, at the same time, contains the flexibility required to make them autonomic. Hence, we focus on developing self-organisation techniques for such agent organisations. Next, we discuss the specifics of the self-organisation method that we seek.

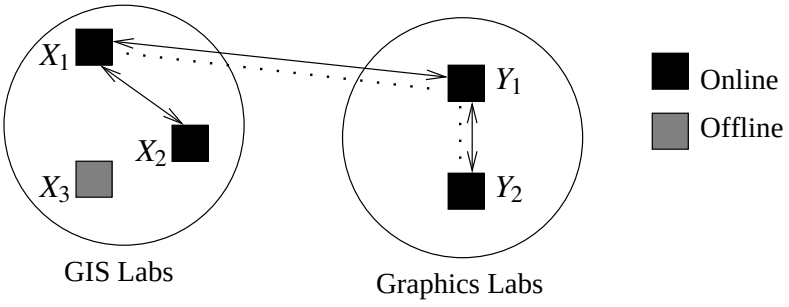
1.2 Decentralised Structural Adaptation

Given the aim of achieving self-organisation in agent organisations, we believe that decentralised structural adaptation provides the most suitable approach (other approaches include modifying the agents sets (Kamboj and Decker, 2006) or their internal characteristics (Klein and Tichy, 2006)). As mentioned before, the structure of an organisation 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.

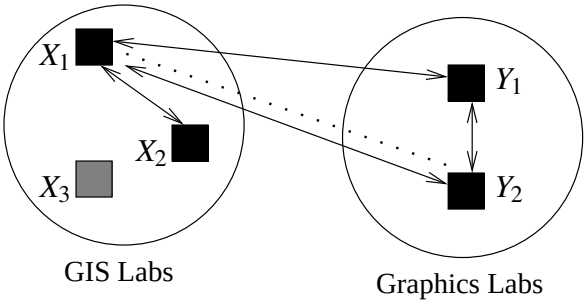
Furthermore, autonomic systems are expected to be deployed in uncertain and changing environments where, in addition to the tasks facing the system, neither the components nor their



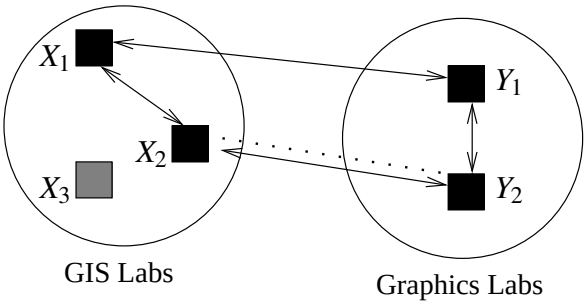
(a) Initial configuration



(b) X_3 goes offline, Y_2 redirects queries



(c) Y_2 and X_1 adapt by forming a relation



(d) Changed structure when Y_2's project changes

FIGURE 1.1: An example of structural adaptation

characteristics 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 organisation, 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 reorganise their interactions to better suit the changed circumstances. For example, consider the earlier described university scenario by focusing only on a few computers in the GIS labs and the graphics labs, as depicted in Figure 1.1(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 1.1(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 realise 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 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 Figure 1.1(c)). With time, that Seoul based project is finished and Y_1 is then being used for a newer project relating to *SaoPaolo*. This GIS information is not present with X_1 but 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 realise that their interactions are not frequent anymore and dissolve their relation, as shown in Figure 1.1(d). Thus, structural adaptation is especially critical in such situations to help the organisation cope with both internal changes and those in the external environment.

Moreover, as the adaptation process itself will require some computation, meta-reasoning 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, considering our earlier example. Y_2 has limited computational resource (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 limited computational resources as well. Thus, 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 meta-reasoning. Now, such meta-reasoning in a multi-agent systems context has been shown to be particularly important for resource-bounded agents in uncertain environments (Raja and Lesser, 2004) because the agents will have to smartly choose how to use their limited resources especially since they cannot predict the future tasks or environmental changes. Thus, it is another 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 seek to develop a novel structural adaptation method for problem-solving agent organisations placed in dynamic environments. Following self-organisation principles, the method should be a decentralised 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. Furthermore, we attempt to develop techniques that can even be applied to scenarios where the agents and/or their internal characteristics are not alterable by the adaptation process. Thus, our mechanism can serve as a self-management tool similar to those envisioned in autonomic systems. We further elucidate our research objectives in the next section.

1.3 Research Requirements

The focus of this thesis is to design a decentralised structural adaptation method for problem-solving agent organisations based on self-organisation principles. This overarching objective entails several requirements including the design of a suitable agent organisation framework along with a performance evaluation mechanism for such organisations, the self-organisation based structural adaptation method, and the demonstration of its effectiveness on various scenarios. Next, we discuss each of these requirements in detail.

1.3.1 Agent Organisation Framework

We seek a problem-solving agent organisation model that serves as a fitting abstract representation of distributed computing systems. Hence, it should provide an appropriate simulation framework for distributed systems by modelling the task environment, the computational entities, and their interactions. Also, in line with our primary objective, the model should focus on the inter-agent interactions; that is, the organisation structure and its effect on the system. Moreover, we also need an evaluation mechanism for the performance of the organisation based on the tasks or goals achieved by it. This method should be such that the critical role played by the organisation structure on the performance is made explicit and clear. In addition, the framework should possess the flexibility to allow agent based adaptation of the organisation. Therefore, any designer of self-organisation techniques, especially those focusing on the structure or network, will be able to see their method in action and evaluate its performance by using this framework, before being transported and put in the actual domain specific autonomic systems. At the same time, we do not require a highly sophisticated model, because this is not the main focus of this thesis, but just enough to satisfy our needs. Moreover, a minimal model is more generic and capable of representing wider range of systems than a sophisticated and specialised model. Towards developing such a generic framework, we seek the representations to be abstract and contain only the essential characteristics. Therefore, the task environment just needs to contain a

stream of tasks requiring some services and the agents should be providing these services using some resources, as these are the basic characteristics of any distributed computing system. Also, since such systems are placed in dynamic environments where it is not possible to predict the future goals or tasks requirements, even our modelled task environment has to be unpredictable with no information available a priori. In addition, since our focus is the agent interactions, the model should contain a basic representation of them using the organisation structure. Finally, a method for measuring an organisation's performance will be useful while evaluating the effectiveness of any adaptation approach. More specifically, the design requirements are as follows:

1. **Task Environment:** Contain a dynamically incoming stream of tasks requiring multiple services with no advance information available about the tasks.
2. **Agent Model:** Suitably represent the individual components of a distributed computing system by accomplishing the incoming tasks through interaction and execution using their limited resources.
3. **Organisational Characteristics:** Govern the agent interactions through the structure and thus, play an important role in the performance of the system.
4. **Performance Evaluation Measures:** Evaluate the performance of the agent organisation in terms of the tasks accomplished and the resources consumed.

1.3.2 Self-Organisation based Adaptation Method

The primary objective of this thesis is to develop a self-organisation based structural adaptation method that can be employed by the agents in a problem-solving agent organisation to improve the performance of the organisation as a whole. Since the purpose of this research is guided by the concept of autonomic systems, the adaptation should target the organisational characteristics like the structure rather than the individual agent characteristics. This is critical because, changing the characteristics and internal configurations of these components may not be possible on all occasions due to physical and accessibility limitations (e.g data-centres located in remote places cannot easily be replicated) and such changes might be beyond the control of the agents or components themselves. Also, as argued earlier, since the computing systems will be placed in dynamic environments, structural adaptation is necessary to manage the system with regards to the changes affecting it. Therefore, a structural adaptation method will be more widely applicable than those involving changes to the particular agent properties. As discussed before, the adaptation method should enable the agents to choose when and how to adapt, especially when placed in the rapidly changing environments. We elaborate on all of these requirements as follows:

1. **Decentralisation:** The adaptation method needs to be agent-based and be employable by any agent, at any level of the organisation, solely on the basis of its limited local view. This makes the adaptation robust without any critical points of failure.

2. **Continuous:** The method should be a perpetual process, striving to improve the organisation's performance continuously. This property is important because the system might be placed in a highly dynamic environment where changes are taking place at any time.

3. **Local Adaptation:** The agents can adapt only locally and change only those elements of the structure that are under their direct influence, that is their own links and relations only. This enhances the robustness of the method because then, no part of the system will be dependent on any others for their management.

4. **Benefit Globally:** Though based on local adaptation by the agents, the method should lead to the benefit of the organisation as a whole.

5. **Meta-Reasoning:** The method should not only enable the agents to decide on how to adapt, but it should also aid them in choosing when to reason about adaptation. Therefore, the agents should be able to smartly decide when to divert resources for reasoning about adaptation and when to use them solely for task completion.

6. **Open Organisations:** The organisation can be *closed*, wherein the agents are always present, or *open*, in which agents might be entering and/or leaving the system as time progresses. These open organisations represent those distributed computing systems in which resources are added or removed with time. Thus, the existing agents in the system should also be able to adapt the structure to deal with these incoming and outgoing agents in the case of open organisations.

7. **Dynamic Organisations:** The organisation can also be either *static*, wherein the internal properties of the agents are fixed, or *dynamic*, in which the these internal properties of the agents might be changing with time. The dynamic organisations representing the distributed systems in which the resources are changed and updated with time. The agents will also have to adapt to these changes in the system.

8. **Varying Task Environment:** The characteristics of the task environment facing the system might also be changing with time and the adaptation method should be effective in these scenarios as well.

1.3.3 Empirical Evaluation

Self-organisation based approaches tend to not have a theoretical foundation because the global product of the several simultaneous local actions cannot be predicted or modelled theoretically. As a consequence, a theoretical approach towards evaluation of self-organisation is not plausible in most cases (unless all the actions and their interdependencies are all modelled mathematically

based on some theoretical model). Hence, the efficacy of self-organisation inspired methods can mainly be evaluated by thorough experimentation. To this end, we also seek an appropriate experimental setup which will let us create the wide-ranging scenarios that the system is expected to face, and then evaluate its performance in those settings. This requirement entails coming up with suitable simulation parameters and methods for comparison. In detail, the following aspects will need to be addressed:

1. **Simulation Parameters:** The empirical evaluation will be based on various simulation scenarios to represent the various cases that an autonomic system might possibly face. Generation of these scenarios will depend on the relevant simulation parameters. These need to be identified and designed for effective experimentation. In particular, these parameters should enable us to model the openness and dynamism of organisations along with the variance in the task environments.

2. **Methods for Comparison:** The utility of the adaptation method can be obtained by comparing its performance with that of the existing state of the art. In addition, a suitable theoretical upper bound is required to place the performance of the method in terms of the limits on the performance that is possible to be achieved by the system.

1.4 Research Contributions

To achieve the above listed goals, we designed an agent organisation framework and used it as the basis to develop a novel structural adaptation method based on the principles of self-organisation. Our adaptation method enables pairs of agents to continuously and locally reevaluate their structural 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. Specifically, they meta-reason for the adaptation by considering the current free resources at their disposal and the number of successful adaptations in the immediate past, along with a randomised approach for choosing which relations are to be considered for adaptation. Furthermore, our method also aids the agents to adapt in *open organisations*. 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 organisations* are able to adapt the structure to the quickly changing circumstances by associating time-decaying weights to the past interactions while calculating utilities.

By so doing, we claim to advance the state of the art in the domain of adaptation mechanisms for agent organisations in the following ways:

1. We present an abstract representation of distributed computing systems by modelling them as a problem-solving agent organisation. Particularly, our framework is the first one to

highlight the organisational structure and its importance in terms of the system's performance, in such a context. Doing this satisfies all the requirements stated in Section 1.3.1.

2. We provide the first structural adaptation mechanism that is completely decentralised and generically applicable to models with a broad range of inter-agent relations. It is also the first self-organisation inspired approach for adaptation in formally specified agent organisations (as opposed to structure-less systems like swarms and ant-colonies). By this, we satisfy the first four requirements stated in Section 1.3.2 as we have developed a decentralised and continuous adaptation method that is applicable locally for the overall benefit of the system.

3. Our method is the first to consider and address the meta-reasoning aspects involved in the adaptation of agent organisations. Particularly, this satisfies the fifth requirement stated in Section 1.3.2 as our method enables the agents to meta-reason by aiding them to decide when and how to utilise resources for reasoning about adaptation.

4. Our adaptation method is the first one suitable for open and dynamic organisations where the agents and their internal characteristics change with time. It is also the first method that adapts to the changing characteristics of the overall task environment as well. Through this, the rest of the requirements stated in Section 1.3.2 are satisfied because the method helps the agents adapt the structure when some agents enter or leave the system, or their properties are changed over time. Also, using the method, the agents are able to maintain their performance even when the characteristics of the task environment are changed over time.

The third major requirement of this thesis, as identified in Section 1.3.3, is satisfied implicitly during the evaluation of the above described adaptation method. In particular, we highlight the simulation parameters necessary for generating various scenarios like open or dynamic systems and also provide a suitable upper bound and other methods for comparison. Overall, these contributions have led to a number of peer-reviewed publications:

- **Kota et al. (2008):** R. Kota, N. Gibbins and N. R. Jennings (2008). Decentralised structural adaptation in agent organisations. In: *Proceedings of the International Workshop on Organised Adaptation in Multi-Agent Systems (OAMAS) at AAMAS'08*, Estoril, Portugal. pp. 1-16.

  This paper presents a preliminary version of the organisation framework and a decentralised structural adaptation method for closed and non-dynamic organisations by ignoring the meta-reasoning aspects (Contribution 2).

- **Kota et al. (2009*c*):** R. Kota, N. Gibbins and N. R. Jennings (2009). Self-Organising Agent Organisations. In: *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '09)*, Budapest, Hungary. pp. 797-804.

The paper extends Kota et al. (2008) by describing a more sophisticated adaptation method which also includes meta-reasoning and strategies for dealing with open organisations (Contributions 2, 3 and part of 4).

- **Kota et al. (2009*b*):** R. Kota, N. Gibbins and N. R. Jennings. A Generic Agent Organisation Framework For Autonomic Systems. In: *1st International Workshop on Agent-Based Social Simulation and Autonomic Systems (ABSS@Autonomics 2009)*, Limassol, Cyprus.

  This paper presents, in detail, our agent organisation framework for representing distributed computing systems and shows how it is suitable for modelling autonomic systems (Contribution 1).

- **Kota et al. (2009*a*):** R. Kota, N. Gibbins and N. R. Jennings. Decentralised Approaches for Self-Adaptation in Agent Organisations. Submitted to: *ACM Transactions on Autonomous and Adaptive Systems*.

  *This is an extended version of Kota et al. (2009*c*) and gives a more detailed description of the adaptation method and also enhances it to deal with dynamic organisations and varying task environments (Contribution 4).*

- **Chapman et al. (2009):** A. C. Chapman, R. A. Micillo, R. Kota and N. R. Jennings (2009). Decentralised Dynamic Task Allocation: A Practical Game–Theoretic Approach. In: *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '09)*, Budapest, Hungary. pp. 915-922.
  This paper was nominated for the AAMAS Pragnesh Jay Modi Best Student Paper Award. Additionally, it has been invited to appear in a special issue of: *The Computer Journal*.

  This paper was produced alongside this research by working on the related problem of dynamic task allocation. It reports on a novel decentralised technique for planning agent schedules in dynamic task allocation problems and demonstrates the approach over the RoboCup Rescue simulation platform. In particular, it formulates the problem as a Markov game and approximates it using a series of static potential games, which are then solved in a decentralised fashion using the Distributed Stochastic Algorithm. However, this work does not form a part of this thesis, because it pertains to a different problem domain and uses a different approach to the one we present here. Nevertheless, it is relevant to this thesis because while the focus of the thesis is to adapt the agent interactions to facilitate better task allocation, this paper focuses on the correlated problem of dynamic task allocation among the sets of agents

Next, we describe the structure of this thesis by outlining the contents of the subsequent chapters.

1.5 Thesis Structure

The remainder of this thesis is structured as follows:

- In Chapter 2, we review the existing literature in the areas of interest which are organisation modelling and adaptation mechanisms. Particularly, we discuss the current techniques for modelling organisations by studying its features individually— task environment, agent modelling, organisational characteristics and performance evaluation measures. Following that, we focus on self-organisation techniques and also other adaptation mechanisms that are relevant in our context.

- In Chapter 3 we present our framework of agent organisation by describing the task environment representation and the organisational characteristics with particular focus on the structure. We also introduce and explain an evaluation mechanism to measure the performance of the organisation.

- Following that, we present the fundamentals of our adaptation method in Chapter 4 by first detailing the constraints and assumptions placed on the organisation model and then describing our decentralised structural adaptation method for static closed organisations. Later in the chapter, we enhance our method so that it is suitable for open and dynamic organisations as well. In addition, we also discuss how this enhanced method is also useful in task environments with changing properties.

- This is then followed by Chapter 5 in which we record the empirical evaluation conducted on the adaptation method by first detailing the comparison methods and the simulation parameters for all the various scenarios and then analysing the obtained results.

- We conclude in Chapter 6 with a summary of our research and an outlook on future work. It is followed by Appendix A which contains additional sets of results to justify the experimental parameters used in Chapter 5. Finally, Appendix B gives a glossary of all the terms used in this thesis.

Chapter 2

Literature Review

This chapter provides an overview of the current research in the fields of organisation modelling and self-organisation in multi-agent systems. The first section surveys the existing work in modelling agent organisations and evaluates their suitability with respect to our first requirement, stated in Section 1.3.1 which refers to developing a framework for problem-solving agent organisations to represent distributed computing systems. The next section presents the motivation and characteristics of self-organisation in multi-agent systems and follows them up with a description and analysis of the various techniques developed in this particular field. Following that, in Section 2.3, we also look at the prevalent adaptation mechanisms designed particularly for agent organisations. Then, the final section summarises the chapter by highlighting the work that provides the point of departure for this study and also drawing attention to the open issues that need to be addressed to meet the requirements laid out in Section 1.3.

2.1 Agent Organisations

As we are seeking to develop an organisation framework that suitably represents distributed computing systems, it should provide an abstract representation of the components of the system, their social interactions and the tasks that they perform along with the environment that they are based in (discussed in Section 1.1). Correspondingly, organisation modelling involves modelling the task environment, the organisational characteristics (structure and norms), the agents and the measures for evaluating the performance of the organisations (as debated in Section 1.3.1). To this end, in the following subsections, we study the current literature in each of these aspects and evaluate them with respect to the requirements outlined in Section 1.3.1.

2.1.1 Modelling Tasks

Typically, agent organisations execute some task(s). Therefore, the tasks of an organisation form an integral part of its description. Specifically, the set of tasks can be considered to be the problem space of the organisation (Carley and Gasser, 1999). Hence, developing a representation model for the tasks is a necessary step in the process of modelling an organisation.

In more detail, the tasks can be atomic or made up of two or more tasks (or subtasks) which, in turn, may be composed of other tasks. The tasks may have dependencies among them, resulting in a temporal ordering of the tasks in the organisation. In this context, Thompson (1967) identifies three kinds of such dependencies — pooled, sequential and reciprocal. Two or more tasks whose results are jointly required to execute another task are said to be in a pooled dependency relation with each other. A sequential dependency exists between tasks if they have to be performed in a particular sequence. Finally, a reciprocal dependency exists if the tasks are mutually dependent on each other and have to be executed at the same time. However, the tasks dependencies as suggested by Thompson have subsequently been interpreted in different ways in different models.

In particular, Jin and Levitt (1996) model the task dependencies in their 'Virtual Design Team (VDT)' closely following Thompson's model. However, they extend the sequential dependency by representing it as a *successor relationship* between the tasks and further classifying it as a finish-to-start successor (if the task can only be started on the completion of the other task comprising the dependency) or a start-to-start successor (if the task can be started after the start of the other task). Similarly, they consider two types of reciprocal dependencies — information-related and work-related. The former is present between two tasks when some information from the execution of one task is required by another and vice versa. The latter is present in those cases where a change in the execution of one task effects the other and vice versa. This representation is particularly useful if the dependencies are to be modelled in detail.

In contrast, in the PCANS model, Krackhardt and Carley (1998) demonstrate that both pooled and reciprocal dependencies, as described by Thompson, can be derived from sequential dependencies. For example, if task 1 is dependent on the completion of tasks 2 and 3, then tasks 2 and 3 share a pooled dependency relationship. But that also means that task 1 is sequentially dependent on task 2 and task 3. Thus, all the tasks that form the sequential dependencies of a particular task are in a pooled dependency. Similarly, two tasks sharing a reciprocal dependency with each other can be broken into smaller tasks which have a series of sequential dependencies. For example, let there be two tasks, 1 and 2, having a reciprocal dependency with each other. They can be divided as, say, task 1a and task 1b (representing the first task) and task 2a and task 2b (representing the second one) such that task 1a is sequentially dependent on task 2a, which is sequentially dependent on task 1b, which, in turn, is sequentially dependent on task 2b. Thus, their representation enables the designer to model just a single dependency.

Other than the task dependencies, the task environment of an organisation can be further characterised on the basis of the degree of repetition, volatility, bias, and complexity (Carley and Gasser, 1999). Thus, tasks could be repetitive, quasi-repetitive (same type of tasks, but some details in the specific instances are different) or non-repetitive. Volatility denotes the rate of change of the tasks. Bias represents the extent to which all possible tasks may have the same outcome, while complexity denotes the amount of processing required by the tasks. Changes in the task environment can also be classified as — sudden change, oscillating, and gradual change. In sudden change, the task environment changes considerably in a short span of time. Oscillating environments are those in which the tasks keep fluctuating between two or more types at fairly regular intervals of time. In gradually changing environments, the change in tasks is uniformly distributed over a considerable period of time.

For the requirements of this work, we just need a simple task model containing dependencies, and hence we will use the PCANS model to represent our tasks. This is because this dependencies model reflects most of the tasks in real-life and the allocation of these dependencies among the components of a distributed system is the driving force behind their social interaction, which is the focus of this thesis. Also, the characteristics of the task environment, as described above, will be considered while generating the input tasks during experimentation. In particular, we use scenarios containing quasi-repetitive tasks and also vary the volatility in some scenarios (see Section 5.1.2).

2.1.2 Modelling Organisational Characteristics

Approaches towards organisational design in multi-agent systems can be considered to be either agent-centric or organisation-centric (Lematre and Excelente, 1998). The former focus on the social characteristics of agents like joint intentions, social commitment, collective goals and so on. Therefore, the organisation is a result of the social behaviour of the agents and is not created explicitly by the designer (Gasser et al., 1988; Cohen and Levesque, 1991). On the other hand, in organisation-centric approaches, the focus of design is the organisation which has some rules or norms which the agents must follow. Thus, the organisational characteristics are *imposed* on the agents. Dignum and Dignum (2005) show that an explicit organisation structure helps in the achievement of the objectives of the organisation as these goals may be wider than an individual agent can perceive. Also, the structure will impose restrictions on the interactions of the agents, thus preventing any combinatorial explosion that might happen in structure-less systems. Due to this, but mainly because our requirements relate primarily to problem-solving agent organisations (see Section 1.3.1), we only study organisations in multi-agent systems whose design is modelled explicitly. This means we exclude those approaches based on agent-centric organisation design, as described earlier. In this context, several models for depicting computational organisations have been developed by researchers in the multi-agent systems community. We examine the main ones by considering their ease at representing real-life distributed computing systems, particularly agent interactions as the social interactions between the agents based on the

structure and their adaptation forms the focus of this thesis. For the same reason, we also study how straightforwardly adaptation methods can be embedded into these organisational models.

- **Opera and OMNI:** The OperA framework (Dignum, 2003) is useful for formal specification of agent societies. Its organisational model specifies the organisational characteristics on the basis of the social structure (role characteristics including skills and relations), interaction structure (agent interactions constitute scenes), normative structure (describing expectations and boundaries for agent behaviour) and communication structure (ontology for knowledge representation and communication language). This work is then extended by the OMNI framework (Vazquez-Salceda et al., 2005) which enables the designer to specify the organisational structure, the interactions between the agents and the normative structure independent of the design of the agents. So, it provides a much broader framework for designing agent organisations. However, in both of these frameworks, the agents are not permitted to modify the organisational characteristics that have been predesigned and, hence, they do not provide a suitable platform for self-organisation, thereby not meeting our requirements.

- **Norms based:** López Y López et al. (2006) present a framework for open agent societies by using a norm based approach. Particularly, they analyse the different properties of the norms required in such open systems to regulate agent behaviour and then discuss how to model a normative multi-agent system. Using the formal definition of the norms in the system, the agents are able to reason about the norms and take decisions regarding their actions with respect to the prevailing norms in the system. While such a model is useful for systems in which independently designed agents possessing self-interested goals need to interact with each other, it is not applicable to our scenario where the agents are cooperative towards the system-wide goals.

- **Islander:** Islander (Sierra et al., 2004) uses the following elements to model the organisation — dialogic framework, scenes, performative structure and norms. The dialogic framework defines the roles that can be adopted by the agents. Every role defines a fixed pattern of behaviour expected from the agent playing that role. The dialogic framework also defines the relationships between the roles and the communication ontology. The activities in the organisation are called scenes and involve instances of interactions between the agents playing the roles. A collection of related scenes form a performative structure. The norms represent the commitments, obligations and rights of the participating agents. All these are defined during design time and cannot be changed during execution. Hence, this model too is not flexible enough to incorporate adaptation.

- **ODML and KB-ORG:** In contrast to the above, ODML (Horling and Lesser, 2008) was developed as a quantitative framework for representing organisations. It uses a mathematical syntax, rather than the commonly used structures and norms representation for denoting organisations. The organisational models produced by ODML can be quantitatively compared against each other for a given set of requirements. Therefore, a search

space of organisation instances can be explored to arrive at the most befitting design. Another work on similar lines is KB-ORG (Sims et al., 2008), which is an automated knowledge based process for designing organisations. KB-ORG searches and prunes the design search space by using a search algorithm that utilises the levels of knowledge provided by the requirements specification. Nevertheless, both of these methods for designing organisations are very complex and require an elaborate specification of the organisational requirements. As noted by the authors themselves, a significant amount of domain knowledge and effort is also required to build the models. However, the method only produces an instantiated organisation but not a generic model like we seek. That is, the method needs to be provided the particular set of requirements (like tasks and goals) to produce an organisation suited to them. Thus, the method is not useful for us as our model is expected to function and adapt in a dynamic environment which is not known before-hand.

- **FORM:** In work that has similar aims to our own, Schillo et al. (2002) aim at an organisation framework that is flexible enough for self-organisation. However, they take a strictly emergent view of self-organisation and focus mainly on the social delegation aspects (gift exchange, voting and so on) in agent organisations. Furthermore, their method specifies a set of organisation models, and the participating agents choose, whether or not, to join such organisations. Therefore, this framework does not inherently aid the development of problem-solving agent organisations.

- **NMAS:** Another work that focuses on developing organisation models that permit reorganisation is by Vazquez and López Y López (2007). They follow a norm based approach for modelling hierarchical agent organisations in which every role has a *position profile* associated with it. This profile is specified in terms of positional norms and an agent can take up a role by changing its own set of norms to conform to these positional norms. Therefore, their model allows the agents to change their organisational roles at run-time. However, the model requires that all positions and norms are specified at the outset itself and, thus, does not allow flexibility in the interactions between the positions as sought by us.

- **OMACS:** A more recently developed work towards models for adaptive organisations is presented by Deloach et al. (2008). Their framework allows for specification of roles, and goals in the organisation. Agents can play a role if they possess the necessary capabilities and an agent playing a role is expected to achieve the goals associated with the roles. Therefore, similar to the NMAS model described above, reorganisation here involves reassigning agents to roles and to goals given changes to the circumstances like modifications to the agent capabilities or the goal set. However, the framework requires advance information and complete specification of all the possible dynamics in the organisation. That is, the framework specification includes functions that specify how well a role can perform towards achieving a goal, how well an agent can fit a role and so on. Thus, reorganisation on such a framework reduces to finding the best assignment given these functions specifying the outcome of any assignment. It is not realistic to assume

that such information will be available at the design stage of the distributed systems that we seek to represent. Therefore, this framework also does not match our requirements.

- **AGR:** An earlier and simpler model is provided by Ferber and Gutknecht (1998) and later extended by Ferber et al. (2003). They define an organisation as *a structural relationship between a collection of agents*. Specifically, a meta-model is presented to describe organisations based on agents, groups and roles (AGR). Agents are part of one or more groups and play specific roles within the groups. These groups are created by the agents and the creating agent assumes the group manager role. The structure of the group identifies all the roles and interactions that can appear in the group. The group structure consists of a tuple containing all the possible roles, an interaction graph specifying all valid interactions between the roles in the group and an interaction language that should be followed by the group. The organisation structure contains the set of groups and the possible interactions between the roles belonging to different groups. Therefore, this model provides a reasonable representation for an organisation containing several groups or departments. While we do not require several groups in our organisation, we conform to their idea of the organisation structure determining the interactions between the members, as this interpretation emphasises the importance of the structure.

- **Moise:** A somewhat similar approach is followed by Moise (Hannoun et al., 2000), which considers an organisation structure as a graph defined by a set of roles, a set of different types of links and a set of groups. A role consists of a set of missions. Here, a mission represents a permitted behaviour in the system and is defined by a set of goals, plans, actions and resources. An agent playing a role must obey some permitted behaviours which are specified by the missions comprising the role. The missions may be viewed as the set of services that should be provided by the agent playing the role. Organisation links are arcs between the roles and represent the interactions between the roles. The model suggests three types of links — communication, authority and acquaintance. Communication links specify the kind of communication that can exist between the roles, the protocols to be followed and the particular missions for which they can be used. Authority links represent the subordination of one role to another along with the context for which it is valid. The context is defined by the missions associated with the link. The acquaintance links of a role specify all the roles about which the agent playing this role can possess information and use in its decision mechanism. The definition of groups contains a set of roles, a set of missions (which is a subset of the combined set of missions of the roles belonging to the group) and a set of links which exist between roles belonging to the group. These ideas of relations and interactions with their corresponding graphs provide a good insight into the influence of a structure on the organisation's performance and is therefore helpful to us. However, the links here have a context associated with them and are valid within that context only. We seek an organisation structure that is not so specific or bounded. Nevertheless, some of the ideas used in this model, especially those relating to the organisation structure will be used while developing our model.

- **VDT:** A slightly different approach is followed by the Virtual Design Team (VDT) framework (see Section 2.1.1). Its purpose is to develop a computational model of real-life project organisations (an organisation involved in completion of a project or set of tasks). It does not use the agent-role paradigm. Instead, the agents are fixed to their duties and are called actors. The organisation structure is composed of two structures — a control structure and a communication structure. The former determines the supervision and authority between the members and specifies who reports to whom, while the latter specifies who can talk to whom. An additional parameter representing the *formalisation* of the organisation determines the frequency of communication between the actors. Evidently, VDT attempts to model a problem-solving organisation, and therefore, very relevant for our requirements. However, it lacks flexibility in the organisation structure, as it only permits purely hierarchical organisations. Therefore, we do not directly use the whole VDT model but only some parts of it (explained later in this section).

To sum up, the OperA, OMNI and Islander frameworks allow for an elaborate specification of an agent organisation and the interactions in it. The designer is expected to specify, in detail, the scenes and the interactions between the agents. These models are most suited for creating organisations in an open environment where external agents can enter, participate and leave the organisation. This is because they provide rigid guidelines for the organisation, which the participating agents need to obey, leading to regulation of the organisation. But they are not suitable for our purposes because the agents are not empowered to modify the organisation and have to abide by predefined structure and norms. They are also not apt for specifying organisation-level goals or activities as these organisations only restrict agent interactions, but do not strive to solve any problems or achieve any goals as such. On the other hand, the models that were developed to permit reorganisation (like the frameworks of Schillo et al. and Vazquez et al.) follow norm based approaches to enable the agents to change between specified roles. Therefore, any possible reorganisation process will be restrained to the few configurations visualised at design time. Moreover, these models are also not very suitable for problem-solving organisations in which the agents are internal to the system and with specified capabilities. The mathematical methodologies like ODML and KB-ORG specify frameworks to come up with efficient organisation designs given a quantifiable set of requirements. However, our set of requirements are too simple to warrant these complex algorithms. Moreover, they produce an instantiated, mathematically defined rigid organisation but not a generic model or framework of an organisation. Fortunately, we see that AGR and Moise organisation models are useful to develop goal driven organisations in which agents are part of the structure and perform some tasks, somewhat like the one we seek. Furthermore, while they provide a qualitative model, VDT models a problem-solving agent organisation quantitatively including the load and costs in the organisation. Given this, as our requirements span more than the features offered by any one of these models, we will choose and incorporate the suitable concepts introduced by all of them (see Section 3.2.2 for the details).

2.1.3 Modelling Agents

An overview of modelling agents in the context of organisations is presented by Carley and Gasser (1999). From this, it is apparent that the modelling of agents varies across different organisation models. In particular, agents in the organisation may be homogeneous or belong to different classes. The agent's knowledge and cognition capability may be quite basic and primitive or highly sophisticated. The agents within the organisation may be selfless and cooperative or selfish and competitive. The abilities of the agents may be represented as a simple vector or as a complex combination of skills, decision strategies, preferences, modes of behaviour and so on. However, as we are more interested in the social interactions between the agents than the internal agent characteristics, a simple agent model with skills and standard behaviour will suffice.

In this context, while all the organisation design approaches described in the previous subsection, with the exception of VDT, leave the agent development to the designer, VDT models the members of the organisation called actors in great detail. The main characteristics of the actors are attention allocation (determines the decision making behaviour of how the actor chooses among several task alternatives) and information processing (determines the skills, capacity and other processing characteristics). This design of agents will be partly used in our organisation model as it meets our requirements for modelling agents in the context of problem-solving organisations. Another concept that we will use is obtained from Gershenson (2007) where the agents are required to perform task assignment but can only address one request per time-step. Thus, we will also make use this of representation of agents possessing limited computational capacities so that efficiency of the agents plays a prominent role in the organisation performance.

2.1.4 Evaluating Organisation Performance

Organisation characteristics play a major role in the performance of the organisation (Galbraith, 1977). Here, the criterion we will use for measuring the performance of the organisation is how well it performs its task (Fox, 1988) as we believe this provides a good indication of the organisation's efficiency. Other evaluation methods include a qualitative comparison of the characteristics of commonly used organisation structures in multi-agent systems as presented by Horling and Lesser (2005). A somewhat quantitative method is presented by Grossi et al. (2006) for evaluating the structures of agent organisations. Using graph theory, they quantify the structural features of the organisation and then suggest how the values thus obtained can be used to analyse a number of properties of organisational structure (like robustness, efficiency and flexibility). However, both the above works are completely independent of the tasks that are being handled by the organisation. As a result, they fail to capture the suitability of a structure according to the task environment that it is situated in. Therefore, both these approaches are not appropriate for our second requirement which refers to developing an evaluation function for an organisation on the basis of the tasks executed by the organisation.

On the other hand, in VDT (see Section 2.1.1), the measure of the performance of the organisation is on the basis of the load on the organisation. The load on the organisation is represented in units of *work volume*, thereby providing a common calibration for different tasks. The total work volume of a task is taken as the sum of the production work volume and coordination work volume. The former is further divided into planned work (which is pre-defined in the task description) and production rework (arising due to exceptions). The production rework and the coordination work depend upon the characteristics of the organisation along with the particular task. Therefore, the resultant load on the organisation is a function of the tasks and the organisational characteristics and acts as an performance indicator. Hence, this approach chosen by VDT is more suitable for our requirements, as it is based on the task load of the organisation, and will be taken into account while designing our evaluation mechanism.

2.2 Self-Organisation in Multi-Agent Systems

The concept of self-organisation is inspired from natural systems which function without any external control and adapt to changes in the environment through spontaneous reorganisation. This self-organising ability makes these natural systems robust to changing environmental conditions, thus enhancing their survivability. In the context of computing systems, self-organisation refers to the process of the system autonomously changing its internal organisation to handle changing requirements and environmental conditions (see Chapter 1 for the formal definition). It may involve creating an organisation from a set of un-organised agents or could mean reorganising an already existing organisation of agents or both, creating an organisation and continuously reorganising it as the environmental conditions vary. Further, self-organisation can be classified (Di Marzo Serugendo et al., 2005*a*) as:

- **Strong self-organising systems** — these function without any explicit central control
- **Weak self-organising systems** — these have a central controller or planner, internal to the system, that supervises the (re-)organisation process.

The concepts of strong self-organisation and emergence are closely coupled. Emergence is a phenomenon in which some properties and structure appear at the macro level which are not present at the micro level (Di Marzo Serugendo et al., 2005*b*, 2006). The structure appearing at the macro level is a result of the actions at the micro level, though no such order is observed at the micro level. Relevant examples include the foraging action of ants and appearance of moving patterns in the game of life (Holland, 1998). Often, self-organisation, when occurring in a decentralised manner through local interactions, is an emergent phenomenon. However, it can exist without emergence and vice versa. For our purposes, we study self-organisation mechanisms, irrespective of whether they are an emergent phenomenon or not. Since strong self-organising systems do not have any central control and hence no single point of failure,

they provide the most suitable paradigm for developing autonomic systems. In particular, they have the following characteristics:

- **No External Control:** The primary characteristic of self-organisation is that there is no external control of any kind. Reorganisation processes are initiated internally and result in changing the internal state of the system. Thus, the system manages itself. Also, the process of self-organisation implies that some kind of order be present in the resultant system after organisation or reorganisation.
- **Dynamic Operation:** A self-organising system is expected to evolve over time. Therefore, self-organisation is a continuous process. The property of self-organisation exists permanently in the system.
- **No Central Control:** Strong self-organising systems have no central authority to guide the reorganisation process. This lends robustness to the system as there is no single point of failure. Often, the organisation emerges through the local interactions of the individual components.

Thus, any strong self-organising approach will have to possess these characteristics. In this context, we see that these properties have been captured by the first two requirements stated in Section 1.3.2. Next, we look at the different mechanisms of developing self-organising systems.

2.2.1 Mechanisms of Self-Organisation

Several approaches have been explored by researchers for developing self-organising multi-agent systems. The different approaches can be classified on the basis of the mechanisms employed by them. Di Marzo Serugendo et al. (2006) identify the following categories:

1. **Direct interaction based mechanisms:** These are based on local interactions and computations of the agents that lead the system to converge to a coherent stable state. The focus is on generating an organisation from disordered agents. It is mainly applicable to the structural aspects of organisation like topological placement and communication of the agents (Mamei et al., 2004).
2. **Stigmergy-based mechanisms:** In a stigmergic process, global system behaviour emerges from the *indirect* interactions of the agents that occur by modifying the environment (Bourjot et al., 2003; Steels, 1990). It is difficult to predict the outcome of self-organisation methods based on these mechanisms as the global behaviour emerges through interactions with the environment.
3. **Reinforcement-based mechanisms:** In reinforcement mechanisms, a reward function catalyses the reorganisation. Agent behaviours are rewarded on the basis of some parameters and, consequently, agents adapt their behaviours to achieve better rewards. Therefore,

self-organisation emerges from the adaptive behaviour of the agents. This mechanism is commonly used to create specialisations and divisions of labour among agents (Mano et al., 2006). Some reinforcement mechanisms, broadly classified as collective intelligence (COIN), follow a distributed reinforcement learning approach (Tumer and Wolpert, 2004). A *collective*, in this context, is a system in which the agents making up the system have private utilities, while the system has a global utility. Such a system is *factored* if increasing any agent's private utility cannot decrease the global utility. Furthermore, the system will have high *learnability* if any agent's actions do not affect the private utilities of the other agents. However, in general, a completely factored system cannot have complete learnability and vice-versa (Wolpert and Tumer, 2001). This is because if the system is perfectly factored and the agent's utility is precisely aligned with the global utility, then such a utility function of the agent will be highly influenced by the other agents that also influence the global utility thereby leading to low learnability. Therefore, achieving the requisite balance between the two is a major challenge.

4. **Cooperation-based mechanisms:** Self-organisation can be achieved through locally cooperative interactions between the agents that modify their behaviour on the basis of their local perceptions resulting in system-wide reorganisation. Locally cooperative interactions refer to agents behaving in such a way that they are benevolent towards other agents in the organisation. Two commonly used mechanisms are Organisational Self Design (OSD) and Adaptive Multi-agent Systems (AMAS). These two methods along with their advantages and drawbacks are discussed, in detail, in Section 2.2.2.

5. **Architecture-based mechanisms:** These mechanisms are based on the architectures or meta-models of the organisation. The commonly used method is *holarchies* which are hierarchies made up of holons. Holons are entities that can exist independently or can join with other holons to form bigger holons. Such holons dynamically altering the holarchy according to changes in the environment, forms the basis of self-organisation (Bongaerts, 1998; Fischer, 2005). Therefore, holon based approaches focus on forming and disbanding groups of agents with a strict hierarchy between the groups. For example, Hilaire et al. (2008) use a holonic architecture for decentralised decision making in the agent system. However, such holarchy based approaches focus on forming and disbanding groups of agents and 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.

The above described mechanisms are not completely disjoint from one another. Rather, the classification is based on the behaviours in the agents that lead to self-organisation. In mechanisms using direct interactions like broadcasts, the interactions are responsible for resulting in self-organisation while in those using stigmergy, it arises from the effects left on the environment. In contrast, in the reinforcement mechanisms, the agents modify their internal behaviour based on the feedback received by their actions, while in cooperation-based mechanisms, the behavioural

change occurs through the observations of the agents about the environment and prevailing situation. Finally, in architecture-based mechanisms, the self-organisation characteristics are built into the model itself.

All these mechanisms have been used on several occasions, in different ways, to develop self-organisation systems. Using a combination of these mechanisms, a number of applications have been developed that use self-organisation in multi-agent systems to address real-life problems like information retrieval, resource allocation and so on (Bernon et al., 2006). These applications employ a variety of techniques, some of which can be put into one of the above mentioned types and some cannot, because they cut across the classifications. Even we will end up using a mixture of these mechanisms. Since we require adaptation in explicitly defined problem-solving organisations, the agents will need to be cooperative and also have direct interactions with each other. Moreover, we intend to apply the mechanism of using past information for adaptation, somewhat similar to reinforcement methods.

In the following, we shall study some of the implementations spanning across the various types of self-organisation mechanisms and discuss their usefulness with respect to our requirements (stated in Section 1.3.2) which refer to developing a decentralised structural adaptation method for agent organisations. For lack of a clear and distinct classification, we divide the different implementations into those containing cooperative agents, containing self-interested agents, inspired from the social domains and those applied in computer networks. We study each of them in turn.

2.2.2 Self-Organisation by Cooperative Agents

One of the earliest multi-agent systems to employ self-organisation was developed by Gasser and Ishida (1991). It uses an organisation self-design (OSD) mechanism to provide the agents with the ability to reorganise themselves. Specifically, OSD uses agent composition and decomposition to restructure the organisation. Gasser and Ishida employ OSD in a problem-solving organisation embedded in an environment. The agents split themselves into two or merge with a neighbour depending upon the changing conditions. The agents decide on the reorganisation actions (merging or splitting) on the basis of heuristic rules which are triggered by changes to the requirements or the environment (Ishida et al., 1992). This work was then extended by Kamboj and Decker (2007) to use a more detailed representation for tasks and resources. However, OSD mechanisms function by spawning and merging agents and would not be suitable in scenarios where the agents cannot be merged or divided. Hence, the OSD mechanism doesn't satisfy our requirements as we aim to develop adaptation techniques that change the structure and interactions within the organisation, but not the agents themselves.

Another self-organising mechanism for cooperative agents is based on Adaptive multi-agent systems (AMAS) theory (Picard and Gleizes, 2002). This theory aims to achieve self-organisation in problem-solving multi-agent systems through the locally cooperative actions of the agents

(Capera, George, Gleizes and Glize, 2003; Capera, Gleizes and Glize, 2003). The agents are supplied with skills, communication, knowledge about other agents and criteria to detect non-cooperative situations (NCS). The NCS are those that are adverse to the organisation. They are classified into three kinds— (i) incomprehensible signals from the environment, (ii) perceived information does not initiate any activity in the agent and (iii) the conclusions are not useful to others. The specific list of NCS needs to be pre-defined at design time by the designer. An agent on perceiving a NCS, tries to return to a cooperative situation through actions selected by its decision mechanism. Therefore, through the socially cooperative behaviour of the agents, an organisation emerges and is maintained. However, this approach relies on the designer being able to identify all possible non-cooperative situations and building the agents so that they handle them locally. Thus, this approach cannot be applied in environments where all the states of the organisation cannot be identified or classified at design time. Therefore, AMAS theory is also not suitable for our purpose as it will not fit into the dynamic environments that we will deal with (look at the requirements stated in Sec 1.3.1). However, the concept of cooperative agents working towards the benefit of the organisation matches the self-organisational goals that we seek. In this context, Sims et al. (2003) show that adaptation among cooperative agents which use a kind of social utility through sharing of values between the agents performs better than self-utility based methods in which the agents do not exchange their information. Therefore, in our method, we will be using a similar joint-utility of the agents rather than isolated utility-measures.

2.2.3 Self-Organisation by Self-Interested Agents

While the approaches discussed in the previous subsection are based on agents that are co-operative in nature, self-organisation can also be present in multi-agent systems made up of self-interested agents. Virtual organisations (VOs) (Norman et al., 2004) are an example of such self-interested agents autonomously organising and reorganising into groups depending upon the circumstances in a market-place. Similarly, Knabe et al. (2003) use a holonic approach (see Section 2.2.1) to develop agents that form and disband virtual enterprises (VEs), according to their trade volume with the other agents. Both VEs and VOs are applicable in open environments wherein independent agents compete to provide services, but cannot be applied to a single problem-solving organisation of agents as they do not aim towards an efficient organisation as a whole. Self-organisation by competitive agents within an organisation is also used by Klein and Tichy (2006) to develop a fault-tolerant multi-agent system. In their case, fault tolerance is achieved through the agents dynamically reconfiguring their task specialisations to obtain better rewards. Therefore, every agent uses a simple decision theoretic approach by estimating the rewards for performing any of the other services and then chooses that service which predicts the highest reward to the agent. This reward function is designed such that when there are more agents than the demand requires, the reward is negative and vice versa. However, this work, as such, is also not suitable for our objectives as it is based on the assumptions that tasks do not have dependencies and that all agents have the ability to perform all services. While the ideas

presented in this work are quite useful to us in terms of the agents reasoning based on rewards, they cannot be directly applied to our scenario as in our case, the primary goal of adaptation is improving the efficiency of the organisation without changing the agents. Nevertheless, we pick the idea of using a decision theoretic approach and use it in our method as it provides a good method of representing and evaluating the action choices faced by the agents.

2.2.4 Self-Organisation inspired from Social Domains

Stigmergy and reinforcement based mechanisms, mainly inspired from biology, have been used in reactive agents to develop self-organising multi-agent systems (Mano et al., 2006). The major problem with these mechanisms is that being emergent, the agent design does not guarantee particular global behaviour. Thus, the connection between local behaviours and global results is difficult to obtain. Therefore, the design of the agents is based on extensive experimentation to arrive at the correct parameters that result in useful global behaviour, thereby, making it an unreliable and lengthy approach.

In more detail, a stigmergic self-organisation 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. Agents need to dynamically allocate tasks to servers that are shared between all the agents. The agents attempt to optimise their task allocations by forecasting the future task load on the servers on the basis of the history of server utilisation, obtained from the completed tasks at those servers. Every agent maintains a set of predictors per server. In every such set, one predictor is anointed as the *active predictor* and is used to forecast the future load on that server. On the basis of the forecasts on each of the servers, the agent chooses the server with the maximum capacity forecasted. Thus, the decision mechanism is based on standard decision theory. Also, using the feedback from the time taken to complete its tasks, the agent evaluates the *active predictor* for each server and switches to a different predictor, if necessary. The various strategies followed by the predictors are fixed at design time, only the method of selecting the active predictor is affected by the agent's history. In this way, efficient resource allocation emerges from the indirect interactions between the agents (as the agents only interact with the servers). Some of the ideas presented in this work, mainly the utilisation of the histories of task allocations and the use of decision theory, will be useful for our adaptation method. The major difference between this work and our requirements is that here, 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-organisation process influences the task allocations on a case-to-case basis, while we require self-organisation at the higher level of agent relations that, in turn, influence the task allocations.

Apart from stigmergic self-organisation seen in the biological domain, self-organisation that is seen in social and economic domains like trust behaviour of humans, gossipping and markets can also be applied to develop self-organising computing systems (Hassas et al., 2006). For

example, the T-MAN protocol (Jelasity and Babaoglu, 2005) uses a gossip based mechanism to construct network topologies. The nodes are modelled as agents and select their neighbours through a ranking function that is based on local messages (gossip). This technique is useful to create an organisation, but not to dynamically adapt it according to changing conditions. Hence, it does not satisfy our criteria.

2.2.5 Self-Organisation in Networks

Network related problems provide a suitable scenario for employing self-organisation techniques and are especially important to us as they also focus on the interactions and structure. In this context, Mills (2007) presents a survey of various self-organisation techniques being used in wireless sensor networks. However, these methods are specific to network problems like query-routing and internal power management and cannot readily be ported to generic optimisation problems in multi-agent systems. Nevertheless, one particular work that is relevant to the current study is by Itao et al. (2002) in which autonomous components provide network services by forming relationships with other components based on a reward mechanism. However, the rewards are based on the feedback provided by the end-user for every query. This kind of direct user feedback cannot be expected in the autonomic domains that we intend to deal with. The neighbour selection problem in self-organising networks can also be tackled by using a machine learning approach (Beverly and Afergan, 2007). However, this method requires running the system initially over the training samples to obtain the model that can then be used to predict neighbour suitability. Therefore, it is not an online approach and not suitable for dynamic environments either.

A more relevant network-based 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-organisation approach for the very different problem of structural adaptation. Nevertheless the usefulness of a self-organisation 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 networks. To this end, Biskupski et al. (2007) survey the existing self-organising methods for such systems by comparing them against their model of agent-based self-organisation. Specifically, their localised mechanisms incorporate concepts of feedback, local evaluation functions and decay. Although our domain is more complex, as it deals with agent organisations (rather than networks) which contain several possible types of relations or links between agents influencing both task allocation and load balancing in the organisation, 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.

After analysing the various self-organisation techniques in multi-agent systems, we find that most of them cannot be applied to explicitly modelled problem-solving organisation of agents. This is because they cannot be incorporated in deliberative agents working towards common goals as are present in such organisations. Those that can be applied either self-organise by creating and deleting the agents (OSD) or by enumerating all the possible scenarios (AMAS). But, our requirement is to develop decentralised adaptation techniques without the addition/deletion of agents in a non-deterministic environment. Now, having studied the various self-organisation techniques for multi-agent systems in general, we turn our attention to our problem domain, that is, problem-solving agent organisations. Particularly, in the following section, we look at the various adaptation mechanisms (irrespective of them being self-organising) developed specifically for agent organisations.

2.3 Adaptation in Agent Organisations

Though the works described previously can be applied to agent organisations, they do not deal with explicitly modelled organisations. On this note, Mathieu et al. (2002) suggest that an adaptation method is important to improve the performance (in terms of costs and task completion times) of organisations, though they do not actually provide such a method. In a similar vein, Dignum et al. (2004) discuss reorganisation in agent organisations by examining and classifying the various motivations for reorganisation and the different kinds of reorganisation possible. They broadly classify reorganisation into two types— (i) behaviour change involving short term behaviour modification of some agents and (ii) structural change involving long term changes in the structure of the organisation. Moreover, they emphasise on the necessity of concretely determining the complete utility of an organisation and its structure, which can thereby indicate the benefits of a given type of reorganisation. Thus, while their suggestions further justify our requirement of needing an evaluation mechanism for the organisations, they do not indicate any possible solutions. Along the same lines, Ashri et al. (2003) discuss how managing the relationships between the agents are important for regulating agent interactions. They further present a method for identifying the inter-agent relationships on the basis of the influence of one's actions over the other's environment. However, in our context, the relations between the agents are explicitly defined by the structure.

Moving on to reorganisation designed for formally modelled organisations, Horling et al. (2001) use the TÆMS (Decker and Lesser, 1993) representation to model an agent organisation and propose a diagnostic subsystem to be incorporated inside the agents. Such a subsystem would help the agents identify deficiencies in the organisation and suggest reorganisation measures. In particular, their proposed architecture has three layers— symptoms, diagnosis and reactions. Symptoms are observations of the environment, the diagnosis layer identifies deficiencies on basis of the symptoms, while the reactions layer suggests suitable measures based on the diagnosis. Though this work provides a means for the agents to detect the need for reorganisation, it does not elaborate on concrete reorganisation steps. A mapping between the diagnosis and the

reactions is assumed. Thus, all the adaptation steps have to be pre-designed which is not always possible. In a similar work, Hoogendoorn et al. (2007) present a formal description of the redesign process of organisations based on the AGR organisation model (see Section 2.1.2). Their work suggests an approach to represent the reorganisation process based on the requirements and goals of the organisation. However, it requires a global view of the organisation and does not explicitly specify how to reorganise either.

However, Zhong and DeLoach (2006) do present a preliminary mechanism for centralised reorganisation on the OMACS framework (see Section 2.1.2). Using that framework, adaptation reduces to changing the assignment between agents, roles and goals. The value of any assignment is provided by a given function (acting as a blackbox) and the algorithm provided to choose the best assignment is exponential on the number of agents, roles and goals. This is not applicable to us as, in our case, we do not assume that the utility of any assignment is available a priori through some function. Moreover, we seek a decentralised mechanism.

Similarly, Hubner et al. (2004) incorporate a controlled reorganisation mechanism into the MOISE framework (see Section 2.1.2). Controlled reorganisation follows a top-down approach in which a group of specialised agents carry out the reorganisation process which includes monitoring the organisation, designing the changes and implementing them. Thus, it is not bottom-up or completely decentralised as only some of the agents have reorganising capability. It also requires designing some agents with complex reorganisation modules. While this work addresses the same problem that we are interested in, the approach does not satisfy our requirements as we are interested in a completely decentralised approach in which all agents have reorganisation ability.

Bou et al. (2006*a*,*b*) also incorporate an adaptation mechanism into the Islander organisation model (see Section 2.1.2). In it, a central authority named an 'autonomic electronic institution' modifies the norms of the organisation to accomplish institution level goals. Thus, this mechanism is centralised and is based purely on modifying the organisational level features like norms, without changing the agents or their relationships (structure). Another centralised mechanism developed by Hoogendoorn (2007) uses a max-flow network based approach to dynamically adapt organisations according to environmental fluctuations. It uses the AGR model (see Section 2.1.2) and specifies a mapping between this model and max-flow networks. In this case, the agents are regarded as nodes and their relationships as links of the network. The task requirements are modelled as the environmental pressure on the organisation which is mapped onto the source-sink paradigm of max-flow networks. The adaptation mechanism is based on identifying and adding capacities to the bottlenecks in the system and duplicating roles and the associated links to improve the max-flow of the system. However, again this approach requires a central authority to carry out the reorganisation. Also, it aims at improving the capacity by adding links and nodes but does not attempt to optimise by removing redundant links or nodes.

Another such graph based approach for reorganisation is presented by Wang and Liang (2006). They represent the organisation structure using three graphs— (i) a role graph denoting the relations between the roles (ii) an agent graph, which is an instantiation of the role graph, depicting the relations between the agents depending on the roles allocated to them and (iii) a connector graph which links the agent graph to the role graph. The reorganisation process is based on graph transformation that occurs as agents shift between the roles. However, this transformation takes place according to predefined changes that correspond to different possible scenarios. Therefore, like AMAS, this method also requires that all the situations are anticipated at design time.

Apart from methods designed specifically for particular agent organisations, as studied above, social networks (Watts, 2001; Jackson and Watts, 2002) also provide a suitable domain to investigate the structural adaptation methods that we seek. Particularly, social networks containing agents as the nodes are somewhat similar to the agent organisational structures that we focus on.

A good example this is seen in the work by Gershenson (2007) who demonstrates a self-organisation approach for the problem of task assignment in agent networks (these comprise a set of agents with some undirected acquaintance links between them). An agent, that receives a task, needs to send out some dependency requests to its neighbouring agents. Once it receives the responses to these dependencies, its task will be complete. In this way, every agent will receive several such dependency requests from its neighbours, which it stores in a queue and solves in a first-come, first-served basis. Furthermore, an agent can respond to only one request per time-step. Therefore, the performance of the network is measured by the number of tasks that are completed. This depends on the time 'wasted' by the agents waiting for the responses to their dependency requests from agents having long queues. The self-organisation process works by first identifying the agent (say A) with the longest queue. Then, among the agents dependent on A, the one with the largest waiting period chooses another agent (one with the shortest queue) to replace A as its neighbour. Therefore, the global knowledge of the queues of every agent is required in this method. This does not conform with our requirements, as in our case, the agents will only possess local information. However, another method that is mentioned in this work, but not expanded, relates to dynamically creating direct links between frequently interacting nodes to 'cut out' the intermediaries and shorten the communication time. This idea, originally presented by Bollen and Heylighen (1996) in the context of the world wide web, will be built on in our adaptation process as we also seek to improve the efficiency of the structure in a similar fashion.

Miralles et al. (2009) use a meta-level approach for adaptation in a P2P network of agents. The agent at the meta-level is in charge of a cluster of agents and collects information about them. It then uses this information for adapting their local structure. Thus, this approach requires special agents and is partially centralised, both of which are in contrast to the self-organisation principles we intend to follow.

In a similar vein, Gaston and desJardins (2005) also focus on agent networks. Their work deals with agent-based rewiring of the links in order to improve dynamic team formation, thus somewhat resembling structural adaptation in organisations. In particular, they present two strategies— (i) an agent probabilistically replaces one of its current neighbours with a neighbour's neighbour, chosen through preferential attachment (the probability of choosing a node is proportional to its degree) and (ii) an agent wanting to change a link requests information from its neighbours about their best performing neighbour and forms the link with the best performer from among all its neighbours' neighbours. 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 strategies 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.

This work was followed up by Glinton et al. (2008) who 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. However, the model still ignored load due to the existing links and the meta-reasoning aspects of adaptation and is also restricted to single link-type between the agents. 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, and leading to a poorer representation. Also, as seen in Section 2.1.2, many organisation models tend to be composed of multiple link-types or relations.

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 neighbours, thereby reorganising the network. Particularly, whenever an agent receives a task request, it proposes a neighbour to that requesting agent who then accepts this proposal with some probability. Similarly, an agent chooses to remove a neighbour based on probability. The selection of the neighbour for these actions is done by utilising the information present in the action policies of the learning mechanism. However, as in the above work, their network supports only one type of link. Moreover, every type of task has its own separate agent network. Hence, the method does not 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 organisation 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 reorganisation process is also ignored.

In the above study of various mechanisms, we observe that none of them consider the meta-reasoning aspect involved in adaptation. As suggested in Section 1, as the adaptation process also requires computation, meta-reasoning is needed by the agents to decide whether and

when to adapt. Meta-reasoning, 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-organisation scenarios. In particular, Conitzer (2008) emphasises that generic meta-reasoning problems tend to be computationally hard and it is more productive to focus on individually solving the particular cases where meta-reasoning is required. With this knowledge, we seek to only solve our particular meta-reasoning problem and thereby our approach may or may not be applicable to other meta-reasoning scenarios.

We see that most of the adaptation techniques developed for organisations are centralised in nature, while we seek a completely decentralised approach in which all the agents are capable of adaptation without having to hold the complete view of the organisation. The few techniques that are decentralised are applicable only to agent networks supporting just one kind of link between the agents. Moreover, these adaptation methods ignore both the load added onto the agents' reasoning process due to their links and the meta-reasoning involved in the adaptation process. Finally, none of the mechanisms discuss how to adapt when the system is open and dynamic with the agents and their properties changing with time.

2.4 Summary

In this chapter, we have studied several existing approaches for modelling agent organisations which include modelling the tasks, the agents and the organisation characteristics (like structure and norms). We then examined some methods for evaluating the performance of an organisation. Next, we provided the definition and features of self-organisation in multi-agent systems. We followed that by enumerating the various mechanisms of self-organisation. Finally, we concluded by providing a survey of the various self-organisation and adaptation techniques that have been developed and used in multi-agent systems and analysed them with respect to our requirements.

To sum up, we find the PCANS model for task representation suitable for our needs and similarly, the agent model of VDT is appropriate for designing the agents. However, for organisation modelling, we will pick up ideas from a number of organisation models like AGR, MOISE and VDT because none of them individually satisfy all our requirements. Particularly, the organisation structure and characteristics will be built on the AGR and MOISE approaches. Similarly, the performance evaluation mechanism used by VDT will be modified to suit our requirements. It should be noted that none of the models present a way of representing the effect of the relations on the performance of an agent and it is an issue that we will be addressing in our framework as our focus is on the organisation efficiency.

On the other hand, most of the self-organisation techniques that we studied do not meet our requirements, as either they cannot be applied to agent organisations (like stigmergy or network related) or they work by modifying the agents. Most of those that reorganise by changing the structure or norms of agent organisations are centralised in nature. But, as already stated, we

want to explore completely decentralised methods for adaptation. The few decentralised mechanisms developed for agent networks are not applicable to organisations and also do not satisfy our requirements stated in Section 1.3.2. Nevertheless, we can still reuse some of the concepts that have been used in the various self-organising systems. Particularly, we will also use the idea of agents being cooperative and sharing utility values with each other. Similarly, the ideas behind the mechanism based on rewards and the history of the agents will be helpful to us while designing these utility functions. In the same vein, we also pick up the idea of using a decision theoretic approach to represent the adaptation. Building on these preliminary ideas and combining them with more problem-specific solutions (that we will be explaining in the specific contexts), we hope to obtain the same kind of structural adaptation as displayed by some of the centralised approaches in agent organisations.

We now move onto our framework for agent organisations and adaptation methods. The next chapter presents our organisation framework in detail, which is then followed by Chapter 4 which describes our adaptation method.

Chapter 3

Agent Organisation Framework

This chapter describes our model of an agent organisation. As mentioned in Section 2.1, this involves modelling the task environment, the agents and the organisational characteristics. Furthermore, it also includes a method for evaluating the performance of an organisation in terms of its efficiency. Therefore, this chapter addresses all the requirements detailed in Section 1.3.1 which refer to developing a framework for problem-solving agent organisations.

In more detail, the purpose of our organisation framework is to serve as a platform for demonstrating adaptation techniques (as argued in Section 1.1). Hence, we seek to develop a minimal organisation model without attempting to include all the possible features that an organisation might have. Specifically, our model of an agent organisation is a problem-solving group of agents situated in a dynamic task environment. By problem-solving agents, we mean agents that receive some input (task), perform some actions on the basis of that input (processing or execution) and return a result. Correspondingly, the task environment presents a continuous dynamic stream of tasks to be performed. This environment also has other parameters, independent of the task stream, which have a bearing on the organisation. These can be considered to be the costs associated with the environment. The task stream and the environmental costs are used as the basis for evaluating the performance of the organisation. So, we proceed by first describing the task environment, then the agent organisation and finally the evaluation mechanism.

Specifically, the next section details our task environment. The following section presents our model of the organisation and its characteristics by describing the agents and the organisation structure. Our mechanism for evaluating the performance of the organisation for a given set of tasks and environmental parameters is explained in the third section. The final section summarises the chapter.

3.1 Task Environment Representation

The task environment contains a continuous dynamic stream of tasks that are to be executed by the organisation. A task can be presented to the organisation at any point of time and the processing of the task must start immediately from that time-step. Thus, the organisation of agents is presented with a dynamic incoming stream of tasks that they should accomplish. In detail, the organisation of agents provides a set of services which is denoted by S. Every task requires a subset of this set of services. Services are the skills or specialised actions that the agents are capable of. We model the tasks as work flows composed of a set of several service instances (SIs) in a precedence order, thereby representing a complex job as expected in autonomic systems. We define a service instance si_i to be a 2-tuple: $\langle s_i, p_i \rangle$ where $s_i \in S$ (i.e. s_i is a member of the services set S) and $p_i \in \mathbb{N}$ denotes the amount of computation required, measured in terms of the processing required to accomplish that SI.

Following the PCANS model of task representation (see Section 2.1.1), we only consider sequential dependencies between the service instances. Thus, the service instances of a task need to be executed following a precedence order or dependency structure, which also ought to be specified in the task representation. This dependency structure is modelled as a tree in which the task execution begins at the root node and flows to the subsequent nodes. The task is deemed complete when all its SIs have been executed in the order, terminating at the leaf nodes. The complete set of tasks is denoted by W and contains individual tasks w_i which are presented to the organisation over time. As we model the task as a tree, every node or SI in the task will only have one parent (according to the properties of trees) and thus can be reached through only one particular SI for allocation. This also means that any SI occurs only once in the task.

Formally, a task w is defined as a tuple containing a set of service instances and a set of dependency links:

$$w = \langle \{si_i \ \in SI_w\}, H_w \rangle \tag{3.1}$$

where SI_w is the set of SIs required by w and H_w is the set of dependency links containing links between the various si_i of the task. These links are directed arcs between any two service instances depicting a sequential dependency from the source to the destination. So an element h_j of H_w is of the form: $h_j = \langle si_a, si_b \rangle$ where si_a and si_b are the service instances at the origin and the destination of the link.

To illustrate our task model, we will use it to represent a sample task possibly faced by the autonomic university grid network system discussed in Section 1.1. Assume that a project involves producing a predictive model of the infrastructure of a given city. Such a task will involve analysing the GIS data of the required city, obtaining the population density of the city over the past years and then using some kind of statistical analysers on this data to estimate the population distribution in the future. It will also involve predicting the changes to the city transport system using the GIS information on this estimated population, and alongside render the map of the city graphically.

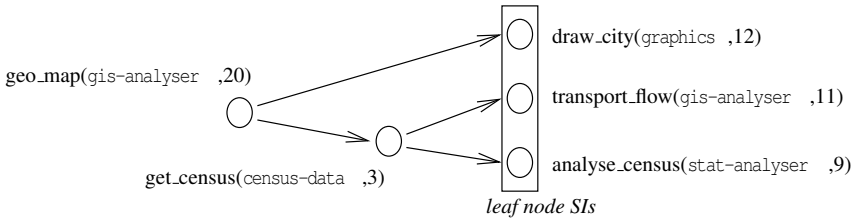


FIGURE 3.1: Representation of an example task

In more detail, let us assume that the first part of this task will be to obtain the geographical data of the city and analyse it. In terms of our model, this can be designated as SI geo_map needing service `gis-analyser` with computation 20 (very intensive job). After this, let us say that the subsequent sub-tasks are obtaining the historical population data of the city and rendering the city-map graphically. These will form SIs get_census and draw_city requiring services `census-data` (getting and cleaning the census information from the archives) and `graphics` (graphically modelling to result in an image). Finally, execution of get_census might reveal that further statistical analysis is required to properly predict the population growth in the future and also that the growth caused by immigration depends on the transport incoming to the city. These can be designated as analyse_census and transport_flow requiring services `stat-analyser` and `gis-analyser` (as the transport network of the city can be obtained by analysing the GIS data) respectively. Also, note that the computation required for geo_map is much higher than that required for transport_flow even though both SIs need the same service. The task structure for this scenario, including the SIs and the dependencies is shown in Figure 3.1. As evident above, we are using the 'sf' font to represent SIs and 'tt' font for denoting services. Now, representing this task formally:

$$\mathsf{model\_city} = \langle \{\mathsf{geo\_map}, \mathsf{get\_census}, \mathsf{draw\_city}, \mathsf{analyse\_census}, \mathsf{transport\_flow}\}, H_{model\_city} \rangle \tag{3.2}$$

where H_{model_city} is the dependency links graph. Thus:

$$H_{model\_city} = \{\langle \mathsf{geo\_map}, \mathsf{get\_census} \rangle, \langle \mathsf{geo\_map}, \mathsf{draw\_map} \rangle, \\ \langle \mathsf{get\_census}, \mathsf{analyse\_census} \rangle, \langle \mathsf{get\_census}, \mathsf{transport\_flow} \rangle\}$$

In summary, our model of the task environment consists of a stream of tasks in which each task is made up of a set of service instances and a set of dependency links between the service instances. Every service instance specifies the service and the amount of computational required. Next, we describe the representation of our agent organisation.

3.2 Organisation Representation

Since we wish the agent organisation to represent a distributed computing system, our organisation framework consists of a set of computational agents representing the individual components. In this context, an agent is an independent computational entity that can provide one or more services. Specifically, we model our agents by simplifying the agent model used by VDT (see Section 2.1.3). In particular, we consider only the information processing characteristics of the agents by overlooking the attention allocation characteristic. The attention allocation characteristic enables an agent to schedule its allocated tasks, however an agent's task scheduling algorithms will depend on the system that is being represented. But, this aspect is internal to an agent and independent of the organisational dynamics which is our primary focus. Therefore, we do not need to model this aspect. Rather, we will simply assume that the agents do not have a choice and have to execute all the service instances allocated to them on a first-come-first-served basis. Next, we describe the agent model in detail, and following that, present the organisation structure and its effect on the allocation mechanism.

3.2.1 Agent Representation

The agents are associated with particular sets of services (like say, in the example university grid system, a computer contains both a GIS data analysing capability and a high end graphics rendering ability, thus containing two services in its service set). These sets can be overlapping; that is, two or more agents may provide the same service. Also, building on the agent model used by Gershenson (see Section 2.1.3), every agent also has a computational capacity associated with it. The computational load on an agent (explained later), in a time-step, cannot exceed this capacity. This modelling of resource constrained agents is necessary because, often the components of an autonomic system can be small embedded devices with low computational power. Also, more generally, all real systems are bounded in their processing capabilities. Formally, let A be the set of agents in the organisation. Every element in this set is a tuple of the form:

$$a_x = \langle S_x, L_x \rangle \tag{3.3}$$

where the first field, $S_x \subseteq S$ denotes a set of services that belong to the complete service set S and $L_x \in \mathbb{N}$ denotes the capacity. The agents, their service sets and their capacities may change during the lifetime of the organisation (see Section 3.2.4 for details).

The other features of an agent organisation, in general, are its structure and norms. The structure of an organisation represents the relationships between the agents in the organisation, while the norms govern the kind of interactions and messages possible between the agents. However, since we are developing a problem-solving organisation, the agents are all internal to the organisation and share the same goals. Moreover, all the agents will be designed in the same way, and therefore, their interaction protocol will be similar and can be internally specified. Therefore,

an explicit definition of norms is not required to regulate their interactions. Thus, in our model, the relationships between the agents (denoted by the structure) also determine the interactions between the agents. Formally, an organisation is defined as consisting of a set of agents and a set of organisational links. It can be represented by a 2-tuple of the form:

$$ORG = \langle A, G \rangle \tag{3.4}$$

where A, as stated above, is the set of agents and G is the set of directed links between the agents (will be described later in this section).

As mentioned previously, the organisation is presented with a continuous stream of tasks which are completed by the agents through their services. Tasks come in at random time-steps and the processing of a task starts as soon as it enters the system. Task processing begins with the assignment of the first SI (root node). 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. 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. This action mechanism of an agent is presented in a pseudocode form in Algorithm 3.1.

```
1  l_x ← 0;                                              // load initialised to zero
2  while l_x ≤ L_x AND W_x^P ≠ ∅ do                      // W_x^P is queue of pending SIs
3      si_i, type ← W_x^P.dequeue();                     // get first element in task queue
4      if type == to_assign then
5          Assign(si_i);
       else
6          Execute(si_i);
7          l_x = l_x + p_i;                              // load increased by executing the SI
8          foreach si_j ∈ si_i.get_dependents() do
9              W_x^P.enqueue(si_j, to_assign);           // enqueue dependent SIs for allocation
           end
       end
   end
```

Algorithm 3.1: Act(): action mechanism of agent a_x in a time-step

We allow the agents to perform more than one action in a time-step to de-couple the time-step of the simulation with the real-time aspect of the actual computing systems. Thus, the time-step of the model places no restrictions whatsoever and can represent one or several processor cycles in the actual system. In contrast, there is no limit to the number of messages that agents can send

or receive in a single time-step. However, the task-related messages received in a particular time-step can only be interpreted by the agent in the next time-step, thus helping to discretise the system.

Next, we present our representation of the organisation structure and follow that with a description of the task allocation mechanism used by the agents.

3.2.2 Organisation Structure

As stated earlier, agents need to interact with one another for the allocation of the service instances. The interactions between the agents are regulated by the structure of the organisation. Inspired from the Moise approach (see Section 2.1.2), we adopt the organisational links paradigm to represent the structure. However, unlike in Moise, the links in our case are not task-specific because we do not assume that the agents will be aware of all the tasks at the outset. Moreover, instead of using several graphs to represent particular aspects like communication, connectivity and so on, we use an *organisation graph* (denoted by G) to represent the structure. The nodes in the graph represent the agents of the organisation while the links represent the relations existing between them. Thus, the structure of the organisation is based on the relations between the agents that influence their interactions.

In more detail, as we use just one organisation graph, it will contain different types of links between the agents to represent the kind of relation present between them (instead of having multiple graphs each with just one type of link, like in Moise). We classify the relationships that can exist between agents into four types — (i) `stranger` (not knowing about the presence), (ii) `acquaintance` (knowing about the presence, but having no interaction), (iii) `peer` (low frequency of interaction) and (iv) `superior-subordinate` (high frequency of interaction). The superior-subordinate relation can also be called the authority relation and depict the authority held by the *superior* agent over the *subordinate* agent in the context of the organisation. The peer relation will be present between agents who are considered equal in authority with respect to each other and is useful to cut across the hierarchy. When two agents are not linked to each other by a relation like acquaintance, peer or superior-subordinate, they are considered to be strangers to each other. Also, the relations are mutual between the agents; that is, for any relation existing between two agents, both the concerned agents respect it. Therefore, even during any adaptation, both the concerned agents will have to agree on changing the relation. The type of relation present between two agents determines the information that they hold about each other and the interactions possible between them. In our context, the information associated with an agent refers to the set of services being provided. Therefore, the service providers that an agent is aware of depends on its relations. More specifically, an agent a_x holds the following information about the services being provided by its relations:

1. For each of its acquaintance a_y, a_x is aware of the service set provided by the acquaintance (S_y of each acquaintance a_y).

2. For each of its peer a_y, a_x is aware of the service set provided by the peer (S_y of each peer a_y).

3. For each of its subordinates a_y, a_x is aware of the accumulated service set provided by the subordinate. The *accumulated service set* 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 sub-graph of agents rooted at its subordinates though it might not know exactly which agent is capable of the service. We denote the accumulated service set of an agent a_y as $AccmSet_y$.

The relations between the agents also help regulate their interactions by determining how agents can allocate tasks to their related agents. This will be clear in the next subsection when we discuss the decision mechanism of the agents. Now, formally denoting the structure, every link g_i belonging to G is of form:

$$g_i = \langle a_x, a_y, type_i \rangle \tag{3.5}$$

where a_x and a_y are agents that the link originates and terminates at, respectively and $type_i$ denotes the type of the link and can take any of the values in the set $\{Acqt, Supr, Peer\}$ to denote the type of relation existing between the two agents. The absence of a link between two agents means that they are `strangers` . In this context, it should be noted that the peer or authority relations supersede the acquaintance relation. That is, when two agents are peers or superior-subordinate, they still count as acquaintances of each other (though the link in G is *Peer* or *Supr* respectively).

To make it clear, we enumerate the properties of the organisation graph:

1. It is a connected graph. That is, the transitive closure of the union of all the relation types is a complete graph.

2. Not more than one link can be present between any two agents. Therefore, a pair of agents cannot have both a peer relation and also an authority relation.

3. There are no self-loops. That is, an agent cannot be a peer or a superior of itself.

4. Peer and acquaintance links are undirected, that is, both the agents will be peers or acquaintances of each other (as the case might be). However, superior-subordinate relation is a directed link, in which one agent is the superior and the other is the subordinate.

5. Superior-subordinate links are acyclic. Therefore, there cannot be a loop of the directed superior-subordinate links.

In this context, it is important to note that we do not discuss the instantiation of the organisation structure. The initial structure is domain-specific and dependent solely on the designer. Nevertheless, when a suitable adaptation process is employed by the agents (like the one we

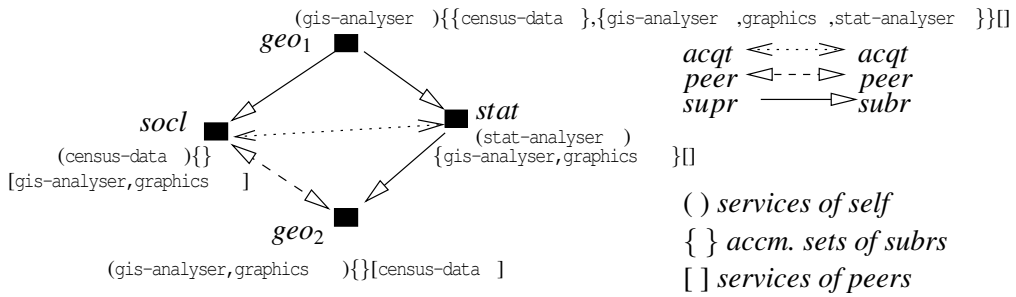


FIGURE 3.2: An example organisation graph

are presenting in this thesis), the initial structure does not hold much significance as it will be quickly modified to suit the task and environmental conditions.

To illustrate the framework, consider a sample agent organisation to represent the autonomic university grid system as described in Section 1.1. Taking a limited view, let us focus on only four agents in this organisation— geo_1 and geo_2 (two computers in the geography department), *socl* (a computer in the sociology department) and *stat* (an analyser in the statistics department). The services provided by the agents are basically their capabilities in terms of hardware, software and data accessible to them (we are denoting agents using italics). Therefore, let us assume that geo_1 provides service `gis-analyser` . Similarly *socl* provides `census-data` , which is the population data of various places in all the past years, and *stat* provides `stat-analyser` service. However, geo_2 is capable of providing both `gis-analyser` (just like geo_1) and `graphics` (because it also contains high end graphics cards for rendering maps). Given this, let us look at the possible structure of the organisation. Let *socl* and geo_2 have a peer relationship. Also, assume geo_1 has two subordinates — *socl* and *stat* (because, say, often GIS based jobs are followed by either census information or statistical analysis). *stat*, in turn, has geo_2 as a subordinate (because statistical analyses sometimes need to represent the results graphically). Moreover, while *socl* and *stat* are acquaintances of each other, geo_2 and geo_1 are not aware of each other. The G for this organisation contains 5 organisational links:

$$G = \{\langle geo_1, socl, Supr\rangle, \langle geo_1, stat, Supr\rangle, \langle stat, geo_2, Supr\rangle, \langle socl, geo_2, Peer\rangle, \langle socl, stat, Acqt\rangle\}$$

For this organisation, the organisation graph is shown in Figure 3.2. The absence of an arrow between two agents means that they are strangers. In addition, the information possessed by the agents about the services provided by their relations is also shown. For example, the accumulated service set (*AccmSet*) of agent geo_1, in turn, contains three sets representing its own service (`gis-analyser`), *AccmSet* of its subordinate *socl* (`census-data`) and of its other subordinate *stat* (`gis-analyser,graphics,stat-analyser`).

Following this description of the organisation structure, we will explain the process followed by an agent for allocating service instances to other agents and how this mechanism is primarily influenced by the structure.

3.2.3 Agent Decision Mechanism

As described above, whenever an agent finishes the execution of a particular SI, it has to allocate each of the subsequent dependent SIs to other agents (this may include itself). This mechanism for allocating SIs to other agents is mainly influenced by the agents' relations. Before, we describe the mechanism, we sum up all the information present with an agent (a_x) about its environment (apart from its own information like service set, capacity and other run-time values):

- The list of all its acquaintances along with their service sets (S_ys).
- The list of all its peers along with their service sets (S_ys).
- The list of all its subordinates along with their respective accumulated service sets ($AccmSet_y$s).
- The list of all its superiors.
- Environmental coefficients (will be explained later).
- Details of every SI allocated to it.
- Details of the dependent SIs of any SI executed by it.

Next, we present the agent's decision mechanism for allocating SIs. While, we use this mechanism for our framework, it could be varied depending on the implementation domain without affecting any other aspect of the framework. In the same vein, it will be evident later that even our adaptation method is not dependent on the allocation mechanism per se. The decision mechanism of an agent for allocating a SI is as follows:

- When an agent needs to allocate a SI, it will allocate the SI to self if it is capable of the service and has no waiting tasks (capacity is not completely used up)
- Otherwise, it will try to assign it to one of its subordinates which contains the service in its accumulated service set. This involves the agent traversing through the accumulated service sets ($AccmSet_y$) of all its subordinates and then randomly choosing one subordinate from among the suitable ones (since the agents do not possess information about the available capacities of the other agents, they do not seek optimal distribution of the load and just allocate randomly among the suitable agents).
- If the agent finds no suitable subordinate (no subordinate or their subordinates are capable of the 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 waiting queue for execution.
- If neither itself nor its subordinates are capable of the service, then the agent tries to assign it one of its peers by traversing through their service sets and choosing from among the suitable ones (those capable of the service). However, in this case, it only checks the set of services directly provided by the peers (since it does not hold information about the accumulated services sets of the peers).

- If none of the peers are capable of the service either, then the agent will pass it back to one of its superiors (who will then have to find some other subordinates or peers to execute the service).
- On the occasions when it does not have any superiors, it checks among its acquaintances for a suitable agent and tries to form a subordinate relation with it, if that doesn't result in a cycle of authority links (a cycle may result in an unending loop of assignment and hence should be avoided). Otherwise, it forms a peer relation with that acquaintance.

The decision mechanism described above is presented in the form of a pseudocode in Algorithm 3.2.

Therefore, an agent mostly delegates SIs to its subordinates and seldom to its peers. Thus, the structure of the organisation influences the allocation of the 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 its relations while allocating a SI. In particular, one unit of management load is added to the load on the agent every time it considers an agent for an allocation (mathematically modelled in Section 3.3). 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. Generally, it is expected that an agent will interact more frequently with its subordinates and superiors than its peers. Also, being cooperative agents that are part of the same organisation, an agent accepts any SIs allocated to it by its superiors or peers. Therefore, the agents are benevolent (Mohamed and Huhns, 2000).

This process of assigning a SI to an agent requires sending and receiving messages to/from that agent. Thus, task allocation also requires inter-agent communication which adds to the cost of the organisation. In more detail, when an agent is assigning a SI, it first sends an *assignment message* with the SI details to the assigned agent. Every assigned agent responds with an *acknowledgement message* comprising a list of the IDs of the agents making up the assigned agents after it, ending with the last assigned agent (which is always the agent that is actually executing the SI). Thus, this *acknowledgement message* is started by the last assigned agent and is passed back, in sequence, by all the assigned agents to finally reach the agent that first initiated the allocation process for that SI. Every message (assignment or acknowledgement) has a communication cost associated with it (explained in Section 3.3). Thus, the structure of the organisation influences the allocation of SIs among the agents and also affects its efficiency. In this section, we have seen that the agents keep track of their environment along with the models of the other agents in the system. Moreover, they are also benevolent towards each other. All these properties justify our use of the agent-paradigm for modelling distributed systems. These characteristics of the agents or components like modelling the other components and benevolence towards them are in line with an agent-based approach.

input: Service instance $si_i = \langle s_i, p_i \rangle$, Agent $a_x = \langle S_x, L_x \rangle$, load of $a_x = l_x$

```
1  if s_i ∈ S_x AND L_x − l_x ≥ p_j then      // if service and capacity is available
2      W_x^P.enqueue (si_i,to_execute);                    // allocate to self
   else
3      listOfSuitableAgents ← ∅;
4      foreach subordinate a_y do
5          if s_i ∈ AccmSet_y then  // AccmSet_y is the accumulated service set of
           a_y
6              listOfSuitableAgents.add( a_y );
           end
       end
7      if listOfSuitableAgents == ∅ AND s_i ∈ S_x then
8          W_x^P.enqueue (si_i,to_execute);      // allocate to self despite filled
           capacity
9          Return() ;
       end
10     if listOfSuitableAgents == ∅ then
11         foreach peer a_y do
12             if s_i ∈ S_y then                    // S_y is the service set of  a_y
13                 listOfSuitableAgents.add( a_y );
               end
           end
       end
14     if listOfSuitableAgents == ∅ then
15         foreach superior a_y do
16             listOfSuitableAgents.add( a_y );
           end
       end
17     if listOfSuitableAgents ≠ ∅ then
18         assignedAgent ← randomly selected from listOfSuitableAgents;
       else
19         foreach acquaintance a_y do          // finding suitable acquaintances
20             if s_i ∈ S_y then
21                 listOfSuitableAgents.add( a_y );
               end
           end
22         assignedAgent ← randomly selected from listOfSuitableAgents;
23         Form_relation_with( assignedAgent);
       end
24     Assign( si_j,assignedAgent);     // 'Assigned' function of    assignedAgent
   end
```

Algorithm 3.2: Assigned (si_i): assignment of a service instance si_i by agent a_x

Next, we illustrate the allocation process and the importance of the organisation structure using the following example:

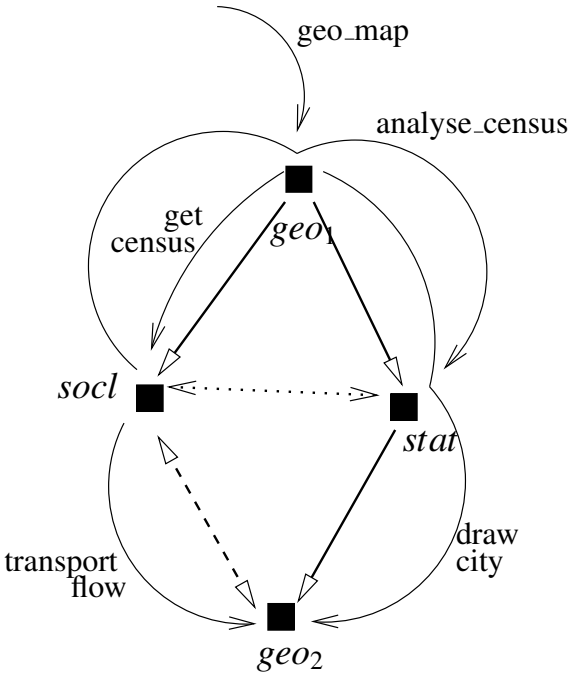


FIGURE 3.3: Distribution of service instances of the task in Figure 3.1 across the agents

As an illustration of the allocation process, consider the sample organisation in Figure 3.2 executing the task shown in Figure 3.1. The allocation of SIs across the agents occurs as shown in Figure 3.3. In detail, we assume that the task arrives at agent geo_1. Hence, geo_1 checks that it is capable of geo_map (as it is capable of service `gis-analyser` and has available capacity) and therefore, allocates geo_map to itself. After execution, geo_1 needs to allocate the two dependencies of geo_map which are get_census and draw_city to capable agents. For allocating get_census, it checks the accumulated service sets of its two subordinates (*socl* and *stat*) and allocates to *socl* (because it is the only one capable of service `census-data`). Similarly, it allocates draw_city to *stat* because this subordinate contains service `graphics` in its accumulated service set. However, *stat* has to reallocate draw_city to its subordinate geo_2 which is actually capable of that service. Similarly, after *socl* executes get_census, it needs to allocate the two dependencies (transport_flow and analyse_census) to appropriate agents. So, *socl* allocates transport_flow to its peer geo_2 as it has no subordinates. It also hands back analyse_census to its superior geo_1 as it has found no suitable subordinates or peers for that service (`stat-analyser`). geo_1 then assigns analyse_census to its subordinate *stat* (capable of service `stat-analyser`) which then proceeds to execute it.

3.2.4 Open and Dynamic Organisations

Given an organisation, the agents in it can remain unchanged over its existence, or they might change with time. For example, new agents might enter the organisation and/or some existing

agents might leave. In this way, the organisation 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 agents in an organisation can be dynamic. More specifically, given an organisation, the set of agents A can remain static over its existence, or agents may join and leave the organisation. The first kind of organisation, in which the set of agents are constant, is here called *closed*, while the latter is called *open*. Another distinguishing feature for organisations is whether the properties of the agents are constant or changing with time. The organisations in which the properties of the agents (their service sets) are not changing are here called *static*, while those in which the agent properties change are referred to as *dynamic*.

Thus an organisation can be closed or open and, in addition, be static or dynamic. Up to now, our description of the organisation model is sufficient to represent closed and static organisations. Thus, in the following, we discuss how to extend it to model open and dynamic organisations.

- **Open Organisations:** For this kind of organisation, the set of agents A changes with time. In particular, we focus on those organisations 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 organisation 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 organisation 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 organisations represent distributed systems in which additional resources might be added to tackle the workload and withdrawn later on (as discussed in Section 1.2).

- **Dynamic Organisations:** In these organisations, 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 $(s_i, s_j \ldots)$ 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 organisations is explained, in detail, in Section 5.1.

In summary, the authority relations impose the hierarchical structure in the organisation, while the peer relations enable the otherwise disconnected agents to interact directly. It is important to note that while we present only these kinds of relations, the model allows the flexibility to depict more relation types in a similar fashion. Thus, the set of the relation types presented here can be expanded or contracted depending on the domain that is to be represented by the organisation model. Using this model, we abstract away the complex interaction problems relating to issues

like service negotiation, trust and coordination. This is because, in our framework, the agents are cooperative and accept each others decisions (on allocations) without conflict and also truthfully share information about their service capabilities. We do so, so that the model keeps the focus on the essence of self-organisation and autonomicity and isolates its impact on system performance.

Having presented our organisational model by describing the agents, including their allocation mechanism, and the organisation structure, we now study our method for evaluating the performance of the organisation.

3.3 Evaluation of Organisation Performance

The performance of a computing system denotes how well it performs its tasks. In terms of an agent organisation, this can be based on the efficiency of the organisation as it performs the tasks (see Section 2.1.4). In terms of our model, the performance measure can be abstracted to the profit obtained by the organisation where the profit is simply the sum of the rewards gained from the completion of tasks when the incurred costs have been subtracted. However, the reward from a task completion depends on how well or how quickly it has been done. This will, in turn, depend on the *load* on the agents, as fully loaded agents will end up having tasks waiting for execution.

In greater detail, our evaluation mechanism is based on the same principles that are used by VDT (see Section 2.1.4). In our context, the load on the agents is of three types— task related (production work in VDT), management related (coordination work in VDT) and adaptation (not modelled in VDT). We further simplify the VDT model by not considering any production rework and denoting coordination work as just the allocation of the SIs in a task. This load on the agents affects the task completion, thereby affecting the rewards obtained, which in turn, determine the profit. The other factor influencing the profit of the organisation is the costs involved in completing the tasks.

In more detail, the cost of the organisation is based on the amount of resources consumed by the agents. In our case, this translates to the cost of sending messages (communication) and the cost of any reorganisation taking place within the organisation. Thus, the cost of the organisation is:

$$cost_{ORG} = C.\sum_{a_x \in A} c_x + D.d \tag{3.6}$$

where C is the communication cost coefficient representing the cost of sending one message between two agents and D is the reorganisation cost coefficient representing the cost of adding or removing a relation. c_x is the number of messages sent by agent a_x and d is the number of relations added or removed in the organisation. Therefore, if the relation between two agents is changed from peer to authority, it would mean dissolving the peer relation and then forming the authority relation, and thus contribute 2 to d. Moreover, we associate costs with the whole organisation rather than the individual agents because these costs may reflect the usage

of resources common to all agents, like the underlying network, instead of the resources at any particular agent.

As stated earlier, agents have limited capacities and their computational load cannot increase beyond this capacity. Since, an agent might perform three kinds of actions in a time-step (task execution, task allocation and adaptation), the load on an agent is the summation of the computational resources used by the three actions and can be represented by three terms. Thus, the load l_x on agent a_x in a given time-step is:

$$l_x = \sum_{si_i \in W_x^E} p_i + M \sum_{si_j \in W_x^F} m_{j,x} + R.r_x \tag{3.7}$$

- p_i is the amount of computation expended by a_x for executing 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 many tasks) executed by a_x in that time-step.
- W_x^F is the set of SIs being allocated by a_x in that time-step.
- M is the 'management load' coefficient denoting the amount of computational units consumed by an agent to consider one of its relations while allocating a single SI.
- R is the 'reorganisation load' coefficient, denoting the amount of computational units consumed by an agent while reasoning about changing a single relation.
- r_x is the number of agents considered by a_x for reorganisation (relation-change), in that time-step.

In this way, M and $m_{j,x}$, together represent the computational load for task allocation that is affected by the relations possessed by the agent, thereby providing a simple and explicit method of denoting the effect of the organisation structure on the individual agents. Similarly, R and r_x are used to represent the load caused by reasoning about adaptation (if any). Thus, they denote the amount of resources at the agent that are diverted for adaptation rather than performing tasks.

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 organisation depend on the speed of completion of tasks. In particular, the maximum possible reward b_w that can be obtained for a task w is the sum of the computation amounts of all its SIs:

$$b_w = \sum_{i=0}^{|SI_w|} p_i \tag{3.8}$$

where SI_w is its set of SIs. If the task is completed within the minimum required time, it would accrue this reward. For delayed tasks, the reward degrades linearly with the extra time taken until it reaches 0:

$$reward_w = b_w - (t_w^{taken} - t_w^{reqd}) \tag{3.9}$$

where t_w^{taken} represents the actual time taken for completing the task, while t_w^{reqd} is the minimum time needed. t_w^{reqd} is given by the depth of the dependency tree of the task w. This is because every child SI can only be executed after the parent SI is executed.

Now, to obtain the maximum possible benefit, the agent should never keep any SIs waiting for either allocation or execution, that is, ideally, none of the agents should ever be overloaded and all the allocations should be a one-step process. Now, the total reward obtained by the organisation is the sum of the rewards of the individual tasks completed by the organisation:

$$reward_{ORG} = \sum_{w \in W} reward_w \tag{3.10}$$

where W is the set of all tasks. Finally, the organisation's performance is measured by the profit obtained:

$$profit_{ORG} = reward_{ORG} - cost_{ORG} \tag{3.11}$$

Thus, for higher profits, the reward should be maximised, while the cost needs to be minimised. Both of these are affected by the allocation of tasks between the agents which, in turn, is influenced by the organisation structure. It is important to note that the agents only have a local view of the organisation. They are not aware of all the tasks coming in to the organisation (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 maximise the organisation profit through faster allocation and execution of tasks. Therefore, by modelling both the decentralisation and individual agent load along with inter-agent dependence and global profit, this evaluation mechanism suitably models the requirements faced by a designer while developing autonomic systems. In the same vein, reasoning and adapting the organisation also take up resources (as denoted by R and D) in our model, thus reflecting real-life scenarios.

3.4 Summary

In this chapter, we introduced our organisation model by presenting our representation of tasks and organisations. The tasks are made up of service instances, each of which specify the particular service and the amount of computation required. The organisation consists of agents providing services and having computational capacities. The relationships between the agents could be just an acquaintance, peer or superior-subordinate. The relations of the agents determine what service information is held by the agents about the other agents and how to allocate service instances to them. We also presented the coefficients that affect the environment (communication cost, reorganisation cost, management load and reorganisation load) and the functions for calculating the costs incurred and the rewards obtained by the organisation on performing the tasks. This enabled us to evaluate the performance of an organisation on the basis of the profit

generated by it. Thus, we addressed the aims of this chapter, which also form all the requirements stated in Section 1.3.1, that involve developing a framework for problem-solving agent organisations.

The next chapter presents our adaptation method by using this organisation framework as the underlying platform.

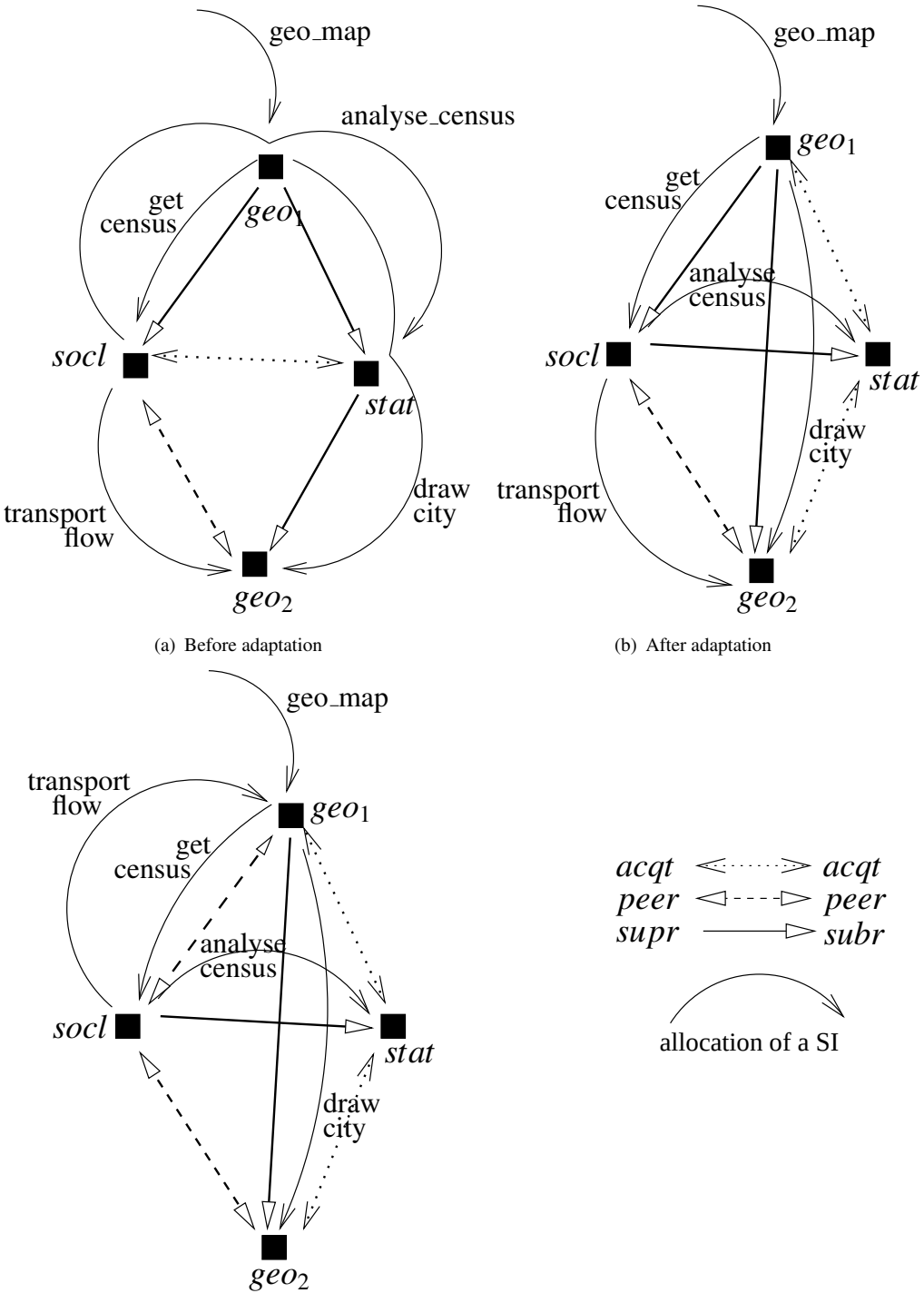
Chapter 4

Decentralised Structural Adaptation

The primary objective of this thesis, as stated in Section 1.3.2, relates to developing a self-organisation based, completely decentralised structural adaptation method for improving the performance of problem-solving agent organisations. In particular, the method should be continuous and employable by any agent in the organisation using only its local view. This chapter attempts to satisfy this requirement by presenting just such an adaptation mechanism. While developing this, we use the framework of agent organisations presented in Chapter 3 as the underlying model.

As argued in Section 1.2, changing the interactions between the agents by modifying the structure is an effective means of adapting the organisation for a better performance. In terms of our organisation model, as agents are performing services and finding other agents to allocate the SIs, they may realise that they have a large number of a particular kind of interaction with some of the agents. Thus, forming interactive relations like authority or peer with these agents (if they are not already present) may make the organisation more efficient by resulting in shorter assignment chains leading to lower cost and load on the agents. Similarly, dissolving existing relations with agents, with whom little or no interaction is taking place, will also improve the efficiency by reducing the management load on the agents as the agents will have fewer relations to consider while allocating SIs.

To make it clearer, we revisit the sample task and organisation scenario presented in Figure 3.3 (reproduced in Figure 4.1(a)). It shows how the allocations of the SIs of a task (shown in Figure 3.1) happens across an organisation (displayed in Figure 3.2). Now, as elaborated in Section 3.2.3, the structure of the organisation influences the allocation of service instances among the agents. Therefore, an efficient structure can lead to a better and faster allocation of tasks. We see that in Figure 4.1(a), the allocation of draw_city and analyse_census was indirect and needed intermediary agents (*stat* and geo_1 respectively). Now, suppose on the basis of some adaptation method, the agents modify their relations to form the structure as shown in Figure 4.1(b). That is, geo_1 and geo_2 decide to form a superior-subordinate relation and so do *socl* and *stat*. Meanwhile *stat* ends up becoming only an acquaintance of geo_1 and geo_2 as they decide to change



(a) Before adaptation

(b) After adaptation

(c) After geo_2 loses `gis-analyser` service

FIGURE 4.1: Allocation of a sample task in the organisation for three structural scenarios

the previously existing authority relations. With this new structure, the allocation of the SIs turns out to be much more efficient as all the allocations end up being direct one-step process. Therefore, they take shorter time because intermediary agents are not involved. Moreover, this allocation process requires less computation and communication because, for any SI, only a single agent performs the allocation and therefore sends only one message. Compared to the previous allocation, this decreases the load on geo_1 and *stat* without putting any additional load on the other agents.

Now, let us suppose that after some time has passed, geo_2 is reconfigured (perhaps, the OS is reinstalled) such that it is no longer able to to provide service `gis-analyser` . In such a scenario, *socl* will no longer be able to delegate transport_flow to geo_2 and will be handing back the SI to its superior geo_1. *socl* does so only after unsuccessfully considering its own subordinates and peers for allocation, thus causing more load onto itself and taking more time as well. Under these changed circumstances, *socl* and geo_1 should realise that it will be better to change their current relation into a peer relation so that *socl* can delegate to geo_1 quicker. A peer relation is better rather than reversing the existing superior-subordinate relation because geo_1 also needs to continue delegating SIs like get_census to *socl*. Hence, these two agents change their relation as shown in Fig. 4.1(c).

In this way, the performance of the organisation can be improved by continually modifying the organisation structure through changes to the agent relationships. This will involve changes to the organisation graph G. Here, while we showed how a more efficient structure can lead to the better allocation of a task, we should note that the organisation is performing several tasks at any given time and that the structure is common to all these possibly dissimilar tasks. Given this, the adaptation method should be such that the agents are able to identify which set of relations will be most suitable for their current context on the basis of the kind of tasks facing them in addition to their own service sets and allocation patterns.

The following section details our work on developing a self-organisation based structural adaptation method that can be employed continually by all the agents in a problem-solving organisation. In particular, we first present decision theoretic modelling of the approach and then discuss the utility functions in Section 4.1.1, before moving onto the meta-reasoning aspects in Section 4.1.2. While this suffices for closed static organisations, further enhancements are required to deal open and dynamic organisations scenarios. Therefore, in Section 4.2, we show how to enhance the fundamental method for such settings. Specifically, we target open organisations in Section 4.2.1 and dynamic organisations in Section 4.2.2. The final section summarises this chapter. With this, we will be attempting to satisfy all the research requirements listed in Section 1.3.2, as these pertain to developing a self-organisation based structural adaptation method for agent organisations.

4.1 Fundamentals of the Adaptation Method

In this section, we present the basics of our adaptation method which is applicable to static closed organisations. The aim of the adaptation method is to determine and effect changes in the organisation structure in order to improve the performance of an organisation. We will be using the framework described earlier, in Chapter 3, as the platform on which to base our method.

Our adaptation method is based on the agents forging and dissolving their peer and authority relations with other agents, thereby modifying the organisation structure (as from our assumption, agents will always maintain the acquaintance relation with the other agents). 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[1]. Thus, we are only focusing on generic adaptation methods that are not dependent on any extra knowledge about the task environment other than the tasks that have already been processed.

Specifically, agents use the information about all their past allocations to evaluate their relations with their subordinates, superiors, peers and acquaintances. This evaluation is based on the possible increase or decrease of the overall load and cost in case the relation had been different (an acquaintance had been a subordinate or a peer, a peer had been a subordinate or only an acquaintance and so on). For example, an agent a_x evaluates its relation with its subordinate a_y on the basis of the number of its SI dependencies that have been executed by a_y. More specifically, a_x assumes that had a_y not been its subordinate, then all its delegations to a_y would have gone indirectly via some intermediary agents. Therefore, a_x will check whether the possible reduction in its own load had a_y not been a subordinate (because one less subordinate will lead to a lower management load during allocations) is more than the possible increase in the load and cost across the organisation (due to the resultant longer allocation chains).

We present the adaptation mechanism, in pseudocode form in Algorithm 4.1, for how an agent a_x should reorganise at a given time-step. The first component (line 1) refers to the meta-reasoning aspect of choosing the particular acquaintances for initiating the reorganisation process. The second component (lines 3–9) explains how it should adapt its relation with one such agent a_y.

The first component, that is the meta-reasoning aspect of adaptation, aids an agent in choosing when to reason about reorganisation and when to solely perform the task related actions, and is described in Section 4.1.2. Now, we start by describing the mechanism that enables an agent to reason and change its relations with its acquaintances (that is, any other agent in the organisation). We formulate this part using a decision theoretic approach since it provides us with a simple and suitable methodology for representing the choices available to the decision maker, thus enabling it to make the right selection (previously justified in Section 2.2). Since, our adaptation method involves agents evaluating and changing their relations, it is befitting to

[1] If there is some information present about the tasks coming in the future, it can be factored into the method. This aspect is further discussed in Section 6.2

```
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
```

Algorithm 4.1: Adaptation algorithm in terms of agent a_x

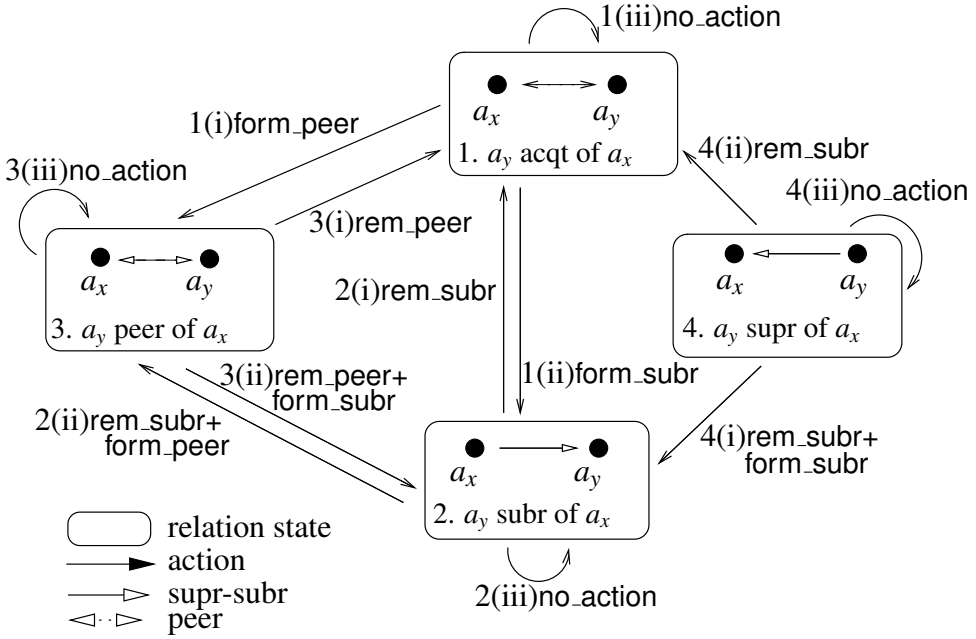


FIGURE 4.2: State transition diagram

represent it in terms of actions and utilities as specified by decision theory. 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, the relation between them has to be in one of the states— purely acquaintance relation, peer relation, superior-subordinate relation or subordinate-superior relation. For each of these 4 states, the possible choices of action available to the agents are:

1. a_y **is an acquaintance of** a_x**:** (i) $\mathsf{form\_peer}_{x,y}$; (ii) $\mathsf{form\_subr}_{x,y}$; (iii) no_action.

2. a_y **is a subordinate of** a_x**:** (i) $\mathsf{rem\_subr}_{x,y}$; (ii) $\mathsf{rem\_subr}_{x,y}$ + $\mathsf{form\_peer}_{x,y}$ (to change to a peer relation); (iii) no_action.

3. a_y **is a peer of** a_x**:** (i) $\mathsf{rem\_peer}_{x,y}$; (ii) $\mathsf{rem\_peer}_{x,y}$ + $\mathsf{form\_subr}_{x,y}$ (to change to a subordinate relation); (iii) no_action.

4. a_y **is a superior of** a_x**:** (i) $\mathsf{rem\_subr}_{y,x}$ + $\mathsf{form\_subr}_{x,y}$; (ii) $\mathsf{rem\_subr}_{y,x}$ + $\mathsf{form\_subr}_{x,z}$ (where a_z is a (indirect) superior[2] of a_y); (iii) no_action.

For example, action 1(ii) ($\mathsf{form\_subr}_{x,y}$) denotes that a_x and a_y take the action of making a_y a subordinate of a_x and undergo 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 are not required. If there were more types of relations in the organisation model, there would be correspondingly more states and transitions to represent them. These possible actions for transitions between the various states are further illustrated by Figure 4.2.

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). Here, form_peer denotes forming a peer relation, while rem_peer denotes removing a peer relation and so on. 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 pre-determined). The utility of performing an action (U_e in line 6) is given by value function V (also called an ordinal utility function (Russell and Norvig, 2003)) associated with the relation.

Since our environment is characterised by various factors like the communication cost and the load on the agents, the value function V will have multiple attributes to represent these different factors. In terms of two agents a_x and a_y jointly deliberating about an action, we list the five attributes that will constitute the value function:

1. change to the load on a_x
2. change to the load on a_y
3. change to the load on other agents of the organisation
4. change to the communication cost of the organisation
5. reorganisation cost incurred by the organisation for taking the action

We selected this set of five attributes because they cover all the factors affecting the organisation's profit (detailed in Section 3.3), but at the same time, can be calculated independent of each other from the history of task allocations. Therefore, this set of attributes exhibits mutual

[2] 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 of authority links in the control graph are not permitted). Hence, making a_z a subordinate entails dissolving its relation with its immediate superior in the authority chain which is a_y.

preferential independence (MPI). That is, while every attribute is important, it does not affect the way the rest of the attributes are compared. Hence, the value function can be represented as simply a sum of the functions pertaining to the individual attributes. That is, it is an additive value function of the form:

$$V = \Delta load_x + \Delta load_y + \Delta load_{OA} + \Delta cost_{comm} + \Delta cost_{reorg} \tag{4.1}$$

In this way, depending on the state, the agents jointly calculate the estimated 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 estimated utility (line 8). Being cooperative, the agents do not have conflicts as the value corresponds to the social utility of the relation to the organisation 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 organisation. The evaluation for the rest of the actions is obtained from Equation 4.1. In the case of the composite actions (rem_subr+form_peer, rem_peer+form_subr or rem_subr+form_subr), 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.

Against this background, we move on to discuss the calculation of the value function for the various actions by detailing how the individual attributes are computed. Following that, in Section 4.1.2, we deal with the meta-reasoning aspects involved.

4.1.1 Value Function Calculation

The value function is the sum of the values of the attributes as shown in Equation 4.1. To represent the performance condition of the organisation (Equation 3.11), a reduction in cost or load (excess load adversely affects the rewards by causing delays to tasks completion) is considered to add positively to the value and vice versa. Table 4.1 lists the calculation of the attributes of the value function for each of the four atomic actions. These calculations use several run-time parameters to take into account the histories of interactions of the agents. Therefore, while we modelled and denoted the general agent properties in Chapter 3, this table uses additional notation to denote these relevant run-time parameters. We describe the notation below:

Assignment: By assignment of a SI to an agent, we mean that the agent has been allocated that SI. It must accomplish that SI by either executing it itself or by assigning it again to some other agent. Formally, when a SI si_i is assigned to agent a_x, then a_x is considered an *assigned agent* for si_i.

Delegation: By delegation of a SI to an agent, we refer to the fact that the agent is executing the particular SI by itself (without reassigning it to someone else). Formally, when an agent a_x

executes a SI si_i, it is considered as the *delegated agent* for si_i. Thus, there could be several assigned agents for a particular SI, but only one delegated agent.

The delegated agent, assigned agents and the assignment chain of a sample SI is illustrated in Figure 4.3.

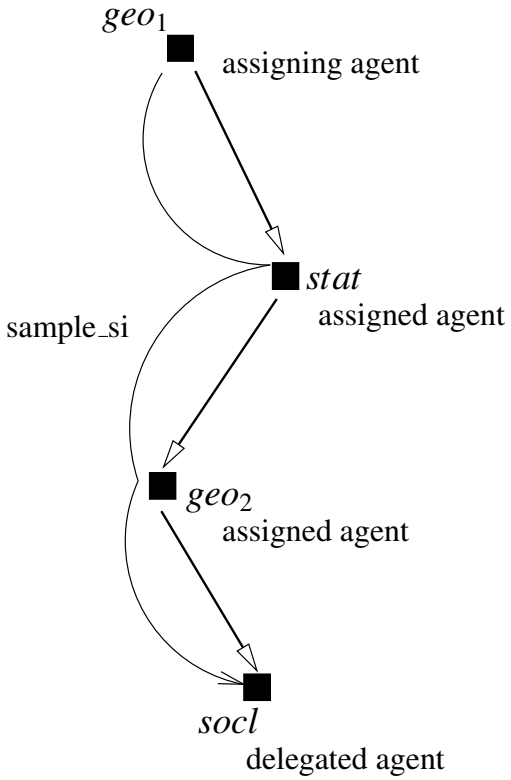


FIGURE 4.3: The assignment chain formed by the allocation of an SI across the organisation

1. $Asg_{x,y}$**:** The number of SIs assigned by an agent a_x to a_y. Assignment of a SI si_i by a_x to a_y means that a_x required that si_i be executed (was assigned to a_x or forms dependency of a SI executed by a_x) and it reallocated si_i to a_y. Thus a_y will have to be a subordinate, peer or a superior of a_x. Also a_y need not necessarily execute si_i itself, it could reassign it to one of its own subordinates, peers or superiors.

2. $Del_{x,y}$**:** The number of SIs delegated by an agent a_x to a_y. Delegation of a SI si_i by a_x to a_y means that a_x is the agent that first required that si_i be executed (as it formed a dependency of a SI executed by a_x) and a_y is the agent that finally executed si_i (that is, a_y is the delegated agent). Note that, a_y may just have an acquaintance relationship with a_x. The delegation is always achieved through one or more assignments.

3. t_x^{tot}**:** The total number of time-steps that a_x has been in existence.

4. $t_{x,y}^{subr}$**:** The number of time-steps that a_x and a_y had a superior-subordinate relation (that is, time-steps that $\langle a_x, a_y, Supr\rangle \in G$) . Likewise, $t_{x,y}^{peer}$ denotes the amount of time that a_x and a_y had a peer relation.

5. $filled_x(t)$: The number of time-steps out of the total time denoted by t that a_x had waiting tasks ($W_x^P \neq \emptyset$; capacity being completely filled by load). The variable t can represent the total time of a_x (t_x^{tot}) or the time duration that a_y was its peer ($t_{x,y}^{peer}$) and so on.

6. $Asg_{x,subr}$**:** The number of SIs that have been assigned by a_x to any of its subordinates. Likewise, $Asg_{x,peer}$ and $Asg_{x,supr}$.

7. $Asg_{x,tot}$**:** The total number of SIs that have been assigned by a_x to other agents. Therefore, $Asg_{x,tot} = Asg_{x,subr} + Asg_{x,peer} + Asg_{x,supr}$.

8. $Asg_{x,y}^{LOAD}$: The management load added onto a_y because of assignments from a_x (the count of these assignments is denoted by $Asg_{x,y}$ as stated above).

9. $IA_{x,y}$: The total number of times, other agents (intermediate agents) were involved in the delegations of SIs by a_x to a_y. Therefore,

$$IA_{x,y} = \sum_{i}^{Del_{x,y}} count_{OA}^{i}$$

   where $count_{OA}^{i}$ is the number of other agents involved in the delegation of si_i from a_x to a_y.

10. $IA_{x,y}^{COST}$: The communication cost due to the delegations from a_x to a_y. For every agent in $IA_{x,y}$, a cost of $2C$ is added because a message is once sent forward and once back. Therefore, $IA_{x,y}^{COST} = IA_{x,y} * 2 * C$.

11. $IA_{x,y}^{LOAD}$: The overall management load put on all the intermediate agents involved in the delegations from a_x to a_y (that is, $Del_{x,y}$). The load values are reported back to a_x along with the assignment information (the assignment message). If an intermediate agent has available capacity (no waiting tasks), it will report a 0 load value for that delegation. Otherwise, the agent will report the actual management load that was put on it due to that SI assignment.

Note that, the *Asg*, *Del* and *IA* values are calculated beginning from the time-instance the concerned agents (a_x and a_y) came into existence. Following this notation, Table 4.1 lists how the five attributes are calculated for each of the four basic actions. The last column in the table denotes which agent will be performing the calculation for that particular attribute.

More specifically, Table 4.1(a) represents the $\mathsf{form\_subr}_{x,y}$ action. When two agents, a_x and a_y, need to evaluate whether forming a superior-subordinate relation will be beneficial, they need to estimate the utility of taking such an action. As stated earlier, this utility is obtained from the value function (Equation 4.1). However the calculation of the attributes, that make up the value function, varies according to the action for which they are being calculated. Therefore, Table 4.1(a) shows how the attributes are calculated for estimating the increase in value by

TABLE 4.1: Attribute functions for the reorganisation actions

(a) Action form_subr$_{x,y}$ between agents a_x and a_y (with a_x as the superior)

	Attribute	Function	Agent
(i)	$\Delta load_x$	$-Asg_{x,tot} * M * filled_x(t_x^{tot})/t_x^{tot}$	a_x
The management load that will be added on a_x due to an additional subordinate. This is estimated by counting all the assignments that a_x had to perform until now and adding a load of M for each of those. This is because, if a_y had been a subordinate of a_x from the beginning, then it will have been considered for each of the assignments of a_x causing an extra load of M for every such instance. This value is then multiplied by a factor which represents how often this increased load will affect the rewards from tasks. This factor is the amount of time-steps that a_x had waiting tasks divided by the total time of its existence. The reasoning is that the increased load will delay the tasks execution only when its capacity is filled.			
(ii)	$\Delta load_y$	$-Asg_{x,y}^{LOAD} * filled_y(t_{x,y}^{subr}) * t_y^{tot}/(t_{x,y}^{subr})^2$ or 0 if $t_{x,y}^{subr} = 0$	a_y
The management load that will be added onto a_y due to possible assignments from a_x. It is estimated by considering the load on a_y due to a_x when it was a subordinate of a_x previously and multiplying it with the fraction of time that a_y had waiting tasks while in the relation and dividing it by the fraction of total time that the relation existed. Thus, the calculated value is normalised to correspond to the total time of the existence and not the relation time.			
(iii)	$\Delta load_{OA}$	$IA_{x,y}^{LOAD}$	a_x
The reduction in the management load on the intermediate agents that had been involved in the delegations to a_y by a_x. As this load value is sent by an intermediate agent only if it has waiting tasks, no time factor like the one in a(i) is required.			
(iv)	$\Delta cost_{comm}$	$IA_{x,y}^{COST}$	a_x
The reduction in communication cost that was associated with the delegations to a_y by a_x.			
(v)	$\Delta cost_{reorg}$	$-D$	constant
This is the reorganisation cost associated with forming or dissolving a relation. Though, it is a one time cost, its affect is spread over the same time window that the rest of the evaluations are valid.			

(b) Action $\mathsf{rem\_subr}_{x,y}$ between agents a_x and a_y (a_x is the superior)

	Attribute	Function	Agent
(i)	$\Delta load_x$	$Asg_{x,tot} * M * filled_x(t^{subr}_{x,y})/t^{subr}_{x,y}$	a_x
The management load on a_x that will be reduced because a_y will no longer be a subordinate. The assignments count for the total time is obtained and then multiplied by the fraction of time that a_x had waiting tasks while in the relation so that the load values correspond to not just the total time but also reflect the factor by which they will affect the task rewards.			
(ii)	$\Delta load_y$	$Asg^{LOAD}_{x,y} * filled_y(t^{subr}_{x,y})/t^{subr}_{x,y} * t^{tot}_y/t^{subr}_{x,y}$	a_y
The management load on a_y that will reduce because it will no longer get direct assignments from a_x. It is multiplied by the fraction of time that it had waiting tasks while in the relationship, similar to a(ii). As in that case, since the load value corresponds to the relation time only, it is divided by the fraction of the time the relation existed so that the calculated value corresponds to total time of existence.			
(iii)	$\Delta load_{OA}$	$-IA^{LOAD}_{x,y} * t^{tot}_x/(t^{tot}_x - t^{subr}_{x,y})$	a_x
The management load that will be put on other agents for delegations to a_y from a_x if it were not a subordinate of a_x. It is estimated by considering the load on intermediate agents, as in a(i), when the relation didn't exist and dividing by the fraction of time that the relation didn't exist so that the load values correspond to the total time.			
(iv)	$\Delta cost_{comm}$	$-IA^{COST}_{x,y} * t^{tot}_x/(t^{tot}_x - n^{subr}_{x,y})$	a_x
The addition to the communication cost associated with the delegation multiplied by the time fraction as in b(iii).			
(v)	$\Delta cost_{reorg}$	$-D$	constant
Similar to a(v).			

taking the $\mathsf{form\_subr}_{x,y}$ action. Similarly, Table 4.1(b) refers to $\mathsf{rem\_subr}_{x,y}$, Table 4.1(c) refers to $\mathsf{form\_peer}_{x,y}$ and Table 4.1(d) refers to $\mathsf{rem\_peer}_{x,y}$ action.

Note that the negative sign, whenever present in the attribute calculations, denotes that the value represents an increase in the load or cost. Also, the attributes are calculated such that the resultant value for any attribute and any action is always normalised to the total time of existence of the two agents involved. Therefore, the utility of one action can be compared directly against that of another action, thus allowing the agents to make a decision. For this reason, the load and cost values are multiplied by time fractions such that the final value always corresponds to this total time. The value function represents the expected change in the load and cost of the organisation if the particular action is taken. Therefore, the intuition behind the attribute calculations is that the past assignments and delegations between the two agents (a_x and a_y) will provide a reasonable indication whether forming or dissolving their relation will reduce the overall load and cost of the organisation and by how much.

Using the value function, every pair of agents jointly evaluates the utility for taking any of the

(c) Action $\mathsf{form\_peer}_{x,y}$ between agents a_x and a_y

	Attribute	Function	Agent
(i)	$\Delta load_x$	$-(Asg_{x,peer}+Asg_{x,supr})*M*filled_x(t_x^{tot})/t_x^{tot}$	a_x
The management load that will be added onto a_x due to an additional peer. It is multiplied by the time factor similar to a(i).			
(ii)	$\Delta load_y$	$-(Asg_{y,peer}+Asg_{y,supr})*M*filled_y(t_y^{tot})/t_y^{tot}$	a_y
Similar to c(i).			
(iii)	$\Delta load_{OA}$	$IA_{x,y}^{LOAD}+IA_{y,x}^{LOAD}$	a_x and a_y
The reduction in the management load on intermediate agents involved in the delegations from a_x to a_y and from a_y to a_x.			
(iv)	$\Delta cost_{comm}$	$IA_{x,y}^{COST}+IA_{y,x}^{COST}$	a_x and a_y
The reduction in the communication cost associated with the delegations in c(iii).			
(v)	$\Delta cost_{reorg}$	$-D$	constant
Similar to a(v).			

(d) Action $\mathsf{rem\_peer}_{x,y}$ between agents a_x and a_y

	Attribute	Function	Agent
(i)	$\Delta load_x$	$(Asg_{x,peer}+Asg_{x,supr})*M*filled_x(t_{x,y}^{peer})/t_{x,y}^{peer}$	a_x
The management load on a_x that will be reduced because a_y will no longer be a peer. Multiplied by the time fraction similar to b(i).			
(ii)	$\Delta load_y$	$(Asg_{y,peer}+Asg_{y,supr})*M*filled_y(n_{y,x}^{peer})/n_{y,x}^{peer}$	a_y
Similar to d(i).			
(iii)	$\Delta load_{OA}$	$-(IA_{x,y}^{LOAD}*t_x^{tot}/(t_x^{tot}-t_{x,y}^{peer})+IA_{y,x}^{LOAD}*t_y^{tot}/(t_y^{tot}-t_{y,x}^{peer}))$	a_x and a_y
The management load that will be added to other agents for delegations from a_x to a_y and from a_y to a_x. It is estimated in the same way as b(iii).			
(iv)	$\Delta cost_{comm}$	$-(IA_{x,y}^{COST}*t_x^{tot}/(t_x^{tot}-t_{x,y}^{peer})+IA_{y,x}^{COST}*t_y^{tot}/(t_y^{tot}-t_{y,x}^{peer}))$	a_x and a_y
The increase to the communication cost because of the delegations. It is estimated in the same way as b(iv).			
(v)	$\Delta cost_{reorg}$	$-D$	constant
Similar to a(v).			

possible actions (depending on their relation) towards changing their relation, at any time-step (though, newly formed relations are allowed to stabilise and are re-evaluated only after a prefixed time interval). Here, joint evaluation means that each agent supplies only some of the attribute values (as evident from the table), but the resultant utility for the action is applicable to both. Being cooperative, the agents do not have conflicts as the value corresponds to the utility of the relation to the organisation and not to the individual agents. Hence, 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.

4.1.2 Meta-Reasoning

We focus now on how an agent can decide which agents to choose for initiating the above detailed adaptation process. In the ideal scenario, at line 1 in Algorithm 4.1, all the related agents (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 meta-reasoning emerges as an important aspect of the adaptation process (this issue has been previously discussed in Section 1.2 and Section 2.3).

In our case, this problem boils down to the following— at any given time-step, an agent should decide on how many and which acquaintances to select for initiating reorganisation 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 randomised 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 related acquaintances (the agents that are peers, superiors, subordinates or just acquaintances) 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 reorganisation deliberation in a given period of time (for 20 relations at an agent, this translates to approximately 80 trials). Therefore, an agent can just randomly choose k agents at a time-step (in line 1). We suggest such a randomised approach as opposed to a sequential selection (where the agent sequentially traverses through its acquaintance list, choosing k agents at a time) because, in general, randomised algorithms are known to provide better solutions for such problems. Moreover, a sequential selection approach will require maintaining additional information about which agents have already been picked, which is not required by our random selection approach for choosing the k acquaintances. Next, even 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 utilised for reorganisation. However, even when the agent is overloaded, reorganisation might be necessary. On such occasions, k is based on the percentage of successful reorganisations in the previous time-step. In more detail, at a time t, k is determined as:

$$k_t = max \begin{cases} 1 \\ (L_x - l_x)/R \\ acqts_x * changed_{x,t-1}/k_{t-1} \end{cases} \tag{4.2}$$

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. As denoted in Section 3.3, L_x is the computational capacity of the agent, l_x is the current load on the agent and R is the reorganisation load coefficient. The minimum possible value of k is limited to 1, so that at least one agent is considered for reorganisation in any time-step. In this way, free capacity is never wasted and, at the same time, an agent will carry out reorganisation even when it has a huge number of pending SIs by adapting k according to its need for adaptation.

We further illustrate the basics of the adaptation method with the help of the following example.

4.1.3 Example

Consider the earlier example of the sample organisation in Figure 3.2 executing the task in Figure 3.1. The allocation of service instances across the agents is reproduced in Figure 4.4 (explained in Section 3.2.3). Also, keeping in line with our assumption, all agents are at least acquainted with each other. Let us assume, for sake of simplicity, that this is the first and only task that has entered the system since the beginning. Also, by time-step 5, the task has been allocated and executed across the organisation as shown. Let us look at agent geo_1 evaluating its relations at time-step 5. Since geo_1 has enough free capacity (and assuming $R = 0.5$), $k_5 = 3$ at step 1 in Algorithm 4.1. For the first case, let a_y in step 2 is $stat$. Therefore, let us look at geo_1 and $stat$ evaluating their relation. According to Figure 4.2, since $stat$ is a subordinate of geo_1, their relation is in state 2 with three possible actions — (i)$rem\_subr_{geo_1,stat}$, (ii) $rem\_subr_{geo_1,stat} + form\_peer_{geo_1,stat}$ and (iii) no_action. Let us first focus on how they calculate the estimated utility of $rem\_subr_{geo_1,stat}$ using Table 4.1(b).

For this example, let the values of the coefficients be:

$$C = 0.25 \qquad and \qquad M = 0.5 \qquad and \qquad D = 0.10$$

Let us assume that all agents existed since the start, that is $t^{tot} = 5$ for all of them. Also, we assume that all the relations existed, as shown, from the beginning. Therefore, the time-period for any of the existing relations (like $t^{subr}_{geo_1,stat}$) is also 5. Moreover, assume that the capacity of agent geo_1 was never filled over these 5 time-steps. Given this, let us look at the first attribute of

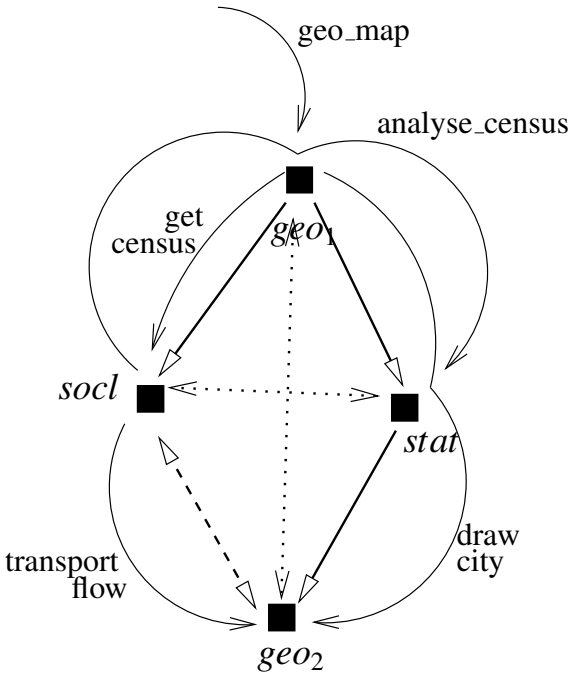


FIGURE 4.4: Allocation and execution of the service instances by the agents

the value function, $\Delta load_{geo_1} = Asg_{geo_1,tot} * M * filled_{geo_1}(t^{subr}_{geo_1,stat})/t^{subr}_{geo_1,stat}$. From the Figure 4.4, we see that geo_1 had to perform three assignments (get_census to *socl*, draw_city to *stat* and analyse_census also to *stat*). Therefore, $Asg_{geo_1,tot} = 3$. However, $filled_{geo_1}(t^{subr}_{geo_1,stat}) = 0$ since geo_1 never had any waiting tasks over this time. Hence, $\Delta load_{geo_1} = 3 * 0.5 * 0 = 0$.

Similarly, looking at the second term $\Delta load_{stat}$, $Asg^{LOAD}_{geo_1,stat} = 0.5$ since *stat* had to perform an assignment (of draw_city) for geo_1 and it took up M load as *stat* had only one subordinate (geo_2). Let us assume that *stat*'s capacity was completely filled for the last time-step (when it started executing analyse_census). Therefore, $filled_{stat}(t^{subr}_{geo_1,socl}) = 1$, while $t^{subr}_{geo_1,stat}$ and t^{tot}_{stat} are 5 each. Hence, $\Delta load_{stat} = 1 * 1/5 * 5/5 = 1/5 = 0.20$

Now, looking at the third term, we find that $IA^{LOAD}_{geo_1,stat} = 0$, since there were no delegations from geo_1 to *stat* (analyse_census which is being executed by *stat* was delegated by *socl* to *stat*, and not geo_1 which had only assigned it). As a result, $\Delta load_{OA} = 0$. Similarly, $\Delta cost_{comm} = 0$ too. Finally, $\Delta cost_{reorg} = -0.10$ as $D = 0.10$.

Having obtained the values of all the five attributes, the sum total of the estimated utility of action $rem\_subr_{geo_1,stat} = 0 + 0.20 + 0 + 0 - 0.10 = 0.10$.

Next, we look at the second possible transition from this state which is $rem\_subr_{geo_1,stat} + form\_peer_{geo_1,stat}$. The evaluation for the *rem_subr* action has already been obtained. So, we focus on the *form_peer* action. The first term $\Delta load_{geo_1} = 0$ since geo_1 had not assigned any service instances to any peers or superiors ($Asg_{geo_1,peer} + Asg_{geo_1,supr} = 0$). Similarly, $\Delta load_{stat} = 0$ as *stat* also did not assign to any peers or superiors. The third and fourth terms, $\Delta load_{OA}$ and $\Delta cost_{comm}$ are also 0 as above because there were no delegations between geo_1 and *stat*

(as described above). Adding all the terms, the estimated utility of action $form\_peer_{geo_1,stat} = 0+0+0+0-0.10 = -0.10$. Therefore the overall utility of the composite action of removing the subordinate and forming the peer is: $0.10+(-0.10) = 0$.

The evaluation for the third possible action (*no_action*) is 0 by default. So comparing the estimated utilities of all the three possible transitions, agents geo_1 and *stat* find that $rem\_subr_{geo_1,stat}$ provides the maximum utility (0.10) and therefore, they take that action. Thus, the superior-subordinate link between geo_1 and *stat* is dissolved. It should be noted that the values calculated here are very low because very little time had elapsed since the beginning of the run and only one task has been executed, and these values will become larger as time goes on and more tasks are processed by the organisation.

Continuing the example, let us consider geo_1 and geo_2 (a_y is now geo_2 at step 2 in the algorithm) evaluating whether they should change their acquaintance relation to a superior-subordinate or a peer relation according to the transitions from state 1 in Figure 4.2. Note that this evaluation is also taking place at time-step 5 as earlier. We first observe their $form\_peer_{geo_1,geo_2}$ evaluation.

Similar to the above *form_peer* calculation, $\Delta load_{geo_1}$ and $\Delta load_{geo_2}$ will be 0 as neither agent has assigned any service instances to peers or superiors. For the $\Delta load_{OA}$ calculation, $IA^{LOAD}_{geo_1,geo_2} = 0$ even though there was a delegation from geo_1 to geo_2 (service instance draw_city). This is because, the only intermediary agent *stat*'s capacity was not completely filled when it performed the delegation. Therefore, it reported a value of 0 to geo_1 along with the acknowledgement message. Also, $IA^{LOAD}_{w,x} = 0$ as geo_2 did not delegate any service instances to geo_1. However, $IA^{COST}_{geo_1,geo_2} = 1*2*C = 1*2*0.25 = 0.50$ because the count of intermediate agents (IA_{geo_1,geo_2}) is 1 (*stat*). As a result, $\Delta cost_{comm} = 0.50$ because the other term, $IA^{COST}_{w,x} = 0$. Overall then, the estimated utility of action $form\_peer_{geo_1,geo_2} = -0-0+0+0.50-0.10 = 0.40$.

Similarly, calculating the estimated utility of the action $form\_subr_{geo_1,geo_2}$ also results in 0.40. The utility of the third possible action, *no_action* is default to 0, so the agents geo_1 and geo_2 can choose arbitrarily between these two actions (in this case, let us say they chose *form_subr*).

Finally, geo_1 also evaluates its relation with *socl* by looking at the possible actions, which are $rem\_subr_{geo_1,socl}$, $rem\_subr_{geo_1,socl} + form\_peer_{geo_1,socl}$ and *no_action*. Their calculation of the attributes, similar to the above, reveals an estimated utility of -0.10 for $rem\_subr_{geo_1,socl}$ and -0.20 for $rem\_subr_{geo_1,socl} + form\_peer_{geo_1,socl}$. Therefore, they chose the third alternative *no_action* which just has an estimated utility of 0. Thus, after geo_1 has re-evaluated all its relations, the structure of the organisation will be as seen in Figure 4.5. The same steps will be followed by the rest of the agents also, depending on their relations.

In this section, we have presented the basics of our adaptation method. Specifically, we described how agents should evaluate the utility of changing their relations and also how to decide when to calculate these utilities. This adaptation is sufficient for closed static organisations. In the next section, we discuss the enhancements needed to tackle open and dynamic organisations.

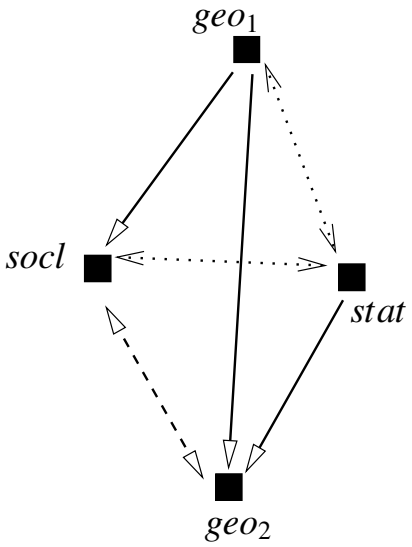


FIGURE 4.5: Organisation structure after geo_1 had reorganised

4.2 Adaptation for Open and Dynamic Organisations

As modelled in Section 3.2.4, organisations can be open and/or dynamic in nature. However, our method, detailed above, might not work well for such organisations. This is because, in open and dynamic organisations, 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, in this section, we extend our adaptation method so that it performs well even in such open and dynamic organisations.

We first describe the extension for open organisations, where some of the agents might be entering and leaving the organisation during its existence. Next, we discuss the modification required for dynamic organisations where the service sets of the agents will be changing with time. Additionally, we explain how this modification also makes our adaptation method suitable for task environments whose characteristics are changing with time.

4.2.1 Open Organisations

When a new agent joins the organisation, 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 reorganise 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-organisation principles (stated in Section 1.3.2). 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 the principle behind 'Win Or Learn Fast' (Bowling and Veloso, 2001) is well suited to our problem. The WoLF principle is— '*learn quickly while losing, slowly while winning*'. In our context, an agent can be considered winning if it has unused capacity and losing otherwise (when it has a pending queue of SIs). Therefore, an agent that is not overloaded will only follow Algorithm 4.1 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 pseudocode in Algorithm 4.2.

```
1 if W_x^P ≠ ∅ then                          // W_x^P is the 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
```

Algorithm 4.2: Algorithm in terms of agent a_x applying WoLF

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. 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. With this technique, 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 organisation. 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 reorganise using the method in Algorithm 4.1 without needing any such additional step. In summary, for open organisations, the adaptation method includes this 'WoLF' principle based algorithm in addition to the fundamental method used for closed organisations for structural adaptation.

Now, we illustrate this extension by continuing the example presented in Section 4.1.3. Let us assume that after a while, a new computer *comp* from the computer science department joins the organisation. It also provides service `graphics` . Now, consider that geo_2 has been receiving a lot of SIs requiring `graphics` . Therefore, its capacity is being filled in the recent time-step and

there is a pending queue of SIs. That is, $W^P_{geo_2}$ contains many SIs requiring service `graphics` . Hence, when geo_2 follows the adaptation algorithm presented in Algorithm 4.2, it reaches step 2 where $s =$ `graphics` . Then, at step 3, the set $A_{graphics}$ contains $comp$ (as it is the only other agent providing `graphics`). So, following steps 4 and 5, it forms a subordinate relation with $comp$ and then start delegating the SIs containing to `graphics` to $comp$.

4.2.2 Dynamic Organisations

As explained in Section 3.2.4, the dynamism of the organisations is caused by the changing service sets of agents. Specifically, 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 the past interactions between the agents. In particular, the agents reevaluate their relations on the basis of estimated utilities which they calculate on the basis of the amount of interactions they had with the other agents since the beginning of their existence. However, this method of using the whole history of interactions as the guidance for adaptation might not be the most suitable approach when the agents' service sets are changing, because the kind of interactions (with the other agents) that they require might also change.

Against this background, we revisit the method that we presented in Section 4.1 so that it remains effective even for these dynamic organisations. An obvious approach to tackle this is by partitioning the estimated utilities on the basis of the services being provided at the agents. In more detail, the terms in Equation 4.1 basically represent the summary of interactions relevant to the two agents concerned (as explained in Section 4.1.1). Now, every individual interaction, that makes up these summaries, is regarding some particular service. Therefore, the interactions can be grouped on the basis of the services. Thus, while an agent calculates the values of the terms in the value function, it will consider only those interactions that were regarding the services that it is currently providing. Therefore, if an agent loses a service, it will ignore its past interactions regarding that particular service for all future utility calculations (if and until it regains the service). Similarly, when an agent gains a new service, it can either start with a '0' as the initial value, or a mean of the values obtained from the other services.

The problem with such an approach is that the interactions between two agents also depend on the other agents in the system. To make our case clear, consider a sample scenario in which agent a_x has been delegating SIs containing service s_i to agent a_y. a_y contains service s_i in its service set S_y, but being generally filled with load, it reallocates the SIs from a_x to its subordinate a_z. Now, if a_y loses service s_i, it will stop including all the interactions with a_x involving s_i and thereby dissolve the relation with a_x. However, nothing should have changed because, in reality, the loss of s_i by a_y would not have affected its interactions at all. Similarly, changes to the service sets of other agents in the system will also affect the interactions between a_x and a_y, but that is not accounted for in this method. Thus, the method of partitioning utilities on the basis of the services assumes that the interactions between any two agents are isolated and independent from the rest of the organisation, and therefore, is not a good solution.

In contrast to the above discussed approach, the extension presented earlier (Section 4.2.1) is somewhat useful for dynamic organisations 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 4.2. 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 not be able to form the best relations for the changed circumstances despite actively seeking specific-service-providing relations using the method.

In this context, a more suitable approach is to give weights to the past agent interactions depending on the time elapsed since they actually took place. Unlike the earlier outlined method of partitioning utilities based on the services, such an approach makes no false assumptions on which particular interactions of the past are relevant and which should be excluded. However, assigning the elapsed-time based weights to the interactions makes the adaptation more responsive to changing scenarios. As recent interactions contribute 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 interactions required will be changing with time.

In more detail, in the fundamental method, the values for the terms in Equation 4.1 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 organisation. A highly dynamic organisation (where agents are losing and gaining services at a fast rate) should have a steeper decay function than an organisation with a slower rate of change.

```
1 Interactions ← set of relevant interactions;
2 total ← 0;
3 time_current ← current time;
4 foreach I ∈ Interactions do
5     total ← total + I.value * Decay(time_current, I.time);
  end
6 return total
```

Algorithm 4.3: Algorithm for calculating the value of a term while evaluating the value function using the decay mechanism

The pseudocode for this method is presented in Algorithm 4.3. In more detail, *Interactions* (line 1) contains the set of all relevant interactions, for that particular attribute, that need to be summarised. While summing them up, each interaction is multiplied by its corresponding

```
   Decay(time_current, I.time)
1  window ← time-window size;
2  if I.time ≤ (time_current − window) then
3      return 0;
   else
4      return (I.time − (time_current − window))/window
   end
```

Algorithm 4.4: Linear decay function within a time window

```
   Decay(time_current, I.time)
1  return e^{−λ∗(time_current − I.time)};        // λ can be adjusted
```

Algorithm 4.5: Exponential decay function

weight which is obtained from the decay function (line 5). We show two sample decay functions in Algorithms 4.4 and 4.5 for organisations changing at a slow and a fast rate respectively. In the linear decay function (Algorithm 4.4), only those interactions that occurred within the given time window are considered and the assigned weights decrease linearly based on how old they are. In the exponential decay function (Algorithm 4.5), the weight value decreases exponentially depending on how old the interactions are. These are two examples of the decay function that can be used for dynamic organisations. The linear decay function will suffice for moderately dynamic organisations while highly dynamic organisations might need exponential decay.

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, unlike the earlier stated approach (partitioning of utilities), it does not make any additional assumptions and is not specifically dependent on the type of changes to the organisation (like the changing service sets of agents). In fact, this approach is useful not only to deal with dynamic organisations, but also for organisations placed in task environments whose characteristics are changing with time. These kinds of environments are detailed and used for empirical evaluation in Section 5.1.2.

In this section, we have presented the extension and modification required for the fundamental method presented in Section 4.1 to function well in open and dynamic organisations. More specifically, for open organisations, the adaptation method is enhanced by including the same principle as behind the WoLF algorithm using which agents that are overloaded will actively seek out suitable relations among the new incoming agents. On the other hand, for dynamic organisations, the fundamental method has to be modified such that weights are associated with the interactions during utility calculation. These weights are highest for most recent interactions and subsequently decay as time elapses. This section is succinctly summarised by Table 4.2.

TABLE 4.2: Mapping of the organisation type to the algorithm required

Type of Organisation	Closed	Open
Static	fundamental method	fundamental method + WoLF principle
Dynamic	fundamental method with decaying weights + WoLF principle	

4.3 Summary

In this chapter, we presented our structural adaptation method using a decision theoretic approach. Using the method, a pair of agents estimate the utility of changing their relation and take the action accordingly. Moreover, our method also enables an agent to meta-reason about when and with whom to initiate reorganisation. We also presented appropriate enhancements to the method to deal with open and dynamic organisations.

In more detail, our adaptation method guides the agent to adapt relations with other agents in the organisation solely on the basis of their history of interactions. Thus, it will result in changes to the organisation structure in an attempt towards improving the performance of the organisation. A particularly useful feature of our method is that it works purely by redirecting agent interactions (via the organisation structure) and does not entail any modifications to the agents themselves or their internal properties. Moreover. as the method only takes into account the past interactions of the agents, it is not dependent on the particulars of the task allocation mechanism used by the agents. Clearly, our method is completely decentralised as it will be adopted by all the agents in the organisation using only their local information. Moreover, it is used by the agents throughout their existence, thus making the adaptation a continuous process. Therefore, our method satisfies the properties of self-organisation that we outlined in Section 2.2. Moreover, the method is also in line with our requirement specified in Section 1.3.2, referring to developing a decentralised agent-based structural adaptation approach. The next chapter explains the experimental conditions used for evaluating our adaptation method and discusses the results obtained.

Chapter 5

Empirical Evaluation

We demonstrate the effectiveness of our self-organisation based adaptation method through empirical evaluation. This chapter discusses the various experiments conducted in this regard. The first section of this chapter describes the experimental design by listing the comparison methods and the various parameters of the simulation. The next section presents and discusses the obtained results. The final section summarises the chapter. This chapter satisfies the third and final set of requirements, stated in Section 1.3.3, which refer to analysing and evaluating the performance of our adaptation method.

5.1 Experimental Setup

The purpose of the experiments is to evaluate the effectiveness of our structural adaptation mechanism developed for problem-solving agent organisations. Therefore, we use the organisation framework described in Chapter 3 as the simulation platform. However, we make the assumption that all the agents are at least acquainted with each other by default. That is, in the organisation graph G, any two agents will have either only an acquaintance link, or a peer link, or an authority link. So, no two agents are strangers and therefore, the graph will be fully connected. As the agents are aware of the service sets of their acquaintances, every agent is aware of the service sets of all the other agents in the system. Additionally, in order to maintain this information, the entry, exit and changes to the service sets of any agent is broadcast to all the other agents in the system. We make these assumptions so that we can focus solely on evaluating the structural adaptation method and isolate it from being affected by any *service discovery* aspects. Moreover, the assumption that all agents are at least acquaintances assures us that, irrespective of the structure, every SI will end up finding some capable agent for its execution as even if an agent has no relations other than acquaintances (therefore, not able to delegate to any subordinates or peers), it would still be able to find a suitable acquaintance to form a relation and then allocate a SI.

Towards evaluating our adaptation method, we compare it, in all its forms, against other intuitive methods. Next, we discuss and justify these comparison methods and follow that up with a description of the simulation parameters.

5.1.1 Methods for Comparison

Since our method has a basic component and other enhancements, we provide names for each of them towards ease of reference:

- **`k-Adapt`:** The fundamental method (without the subsequent enhancements) presented in Section 4.1.
- **`wolf-k-Adapt`:** The extension to the fundamental method, based on the WoLF principle, as discussed in Section 4.2.1.
- **`decay-Adapt`:** The modified method, using the decaying weights technique, presented in Section 4.2.2.
- **`wolf-decay-Adapt`:** This method incorporates both the WoLF principle and the decay mechanism, that is Sections 4.2.1 and 4.2.2 respectively.

Note that the decay function used by `decay-Adapt` and `wolf-decay-Adapt` is linear (presented in Algorithm 4.4). We have conducted experiments by using the exponential decay function (Algorithm 4.5) and found the resulting trends to be broadly similar (these results are presented as part of Appendix in Section A.4); this is because our simulations have low time durations, there is not a marked difference between the linear and exponential functions when the other parameters (window size and decay constant) are adjusted properly to suit the environment. Now, to determine the effectiveness of our approach, we compare its performance with two other intuitive methods— `Central` and `Random` , that act as the benchmarks. An omniscient centralised approach, although impractical, will gives us an indication of the performance that can be aimed at, thereby acting as the upper bound. Similarly, a random approach represents a simple naive solution and should be the lower bound for a smart method like ours. We also 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 below:

- **`Central`:** This is a centralised allocation mechanism containing a central repository that maintains information about the service sets and loads of all the agents in the organisation, and is accessible without cost to any agent. The agents do not need to maintain any relations; whenever an agent needs to allocate a SI, it looks up the repository seeking the most suitable agent (capable of the service and having maximum free capacity, or smallest pending tasks queue, 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. Also,

the allocations lead to the best load balancing across the organisation as the every SI is allocated to the agent with most available capacity. Hence, this method gives an upper bound on the performance of an organisation, but is not a practical or robust solution to the problem 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 4.1), this strategy involves an agent randomly choosing some of its acquaintances for adaptation (line 1), and then randomly choosing a reorganisation action (line 8). For a fair comparison, the rate of change is adjusted so that the number of relations in the structure is roughly equal to that produced by our method. The number of relations in the organisation is maintained at a moderate level through the probability of forming or dissolving a relation. This probability value for an agent is inversely proportional to the existing number of relations of that kind at that agent. Therefore, an agent with very few subordinates has a higher rate of forming an additional superior-subordinate relation than an agent with more subordinates and vice versa. This also makes sure that the performance of `Random` is not affected due to an aggregation of reorganisation cost. Thus, this method represents a random structural adaptation strategy which does not involve any reasoning and constitutes the lower bound.

- **`free-Adapt`:** For this method, the reorganisation load coefficient R is always set to 0. It represents the case when the reorganisation 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 reorganisation instead of just k. This makes it a theoretical upper bound for the performance of `k-Adapt` . In the case of open organisations, `free-Adapt` also includes the WoLF component and therefore, represents the upper bound for `wolf-k-Adapt` .

- **`all-Adapt`:** Same as `free-Adapt` , that is, all acquaintances are chosen for reorganisation in a time-step. However, unlike in `free-Adapt` , reorganisation is not resource-free. Therefore, R is set to the same value as in `k-Adapt` and not 0.

Agents in the `Random` , `free-Adapt` and `all-Adapt` methods follow the same algorithm for task allocation as detailed in Section 3.2.3, while those in `Central` follow the central repository look-up method detailed above. However, the agents in all the procedures are similar in all the other aspects (like task execution, communication, capacities and so on). We evaluate the effectiveness of the methods on the basis of the performance of the organisations employing the methods. The performance of an organisation is determined by the profit obtained by the organisation ($profit_{ORG}$ in Equation 3.11) for a simulation run. We only pick this measure because it encompasses all aspects of the organisation including its costs, allocation efficiency and load balancing (detailed in Section 3.3). Thus, $profit_{ORG}$ is the experimental data variable of interest or the dependent variable. However, there are many other variables that need to be assigned values before running the experiments. Next, we discuss the values that are assigned to these parameters of the simulations.

5.1.2 Simulation Parameters

Simulation parameters are the attributes of the environment which are set according to the experiments. In our case, the simulation parameters include the set of services, set of agents, set of tasks, the initial organisation structure (at the beginning of simulation), the time period of the simulation and the environmental coefficients. While some of these parameters are control variables and assigned a constant value for all simulations, the others are independent variables and are varied across simulations. More specifically Table 5.1 lists the constant values or the fixed ranges assigned to some of the simulation parameters. These are the control variables and the values listed in this table are applicable to all the simulations conducted. However, there are other parameters that are varied across simulations to observe their effect on the organisation's performance and the adaptation methods. These form the independent variables for our experiments. In particular, these relate to the degree of similarity of agents in the organisation and the degree of similarity of the tasks being faced by the organisation. We only look at varying the characteristics of the agent set and the task set because they are the only two main aspects of the system. These two independent variables are discussed next.

5.1.2.1 Distribution of services across agents

The degree of heterogeneity of the agents in the organisation depends on the distribution of services across them. It is a relevant parameter because the significance of the organisation structure is greater when the agents are heterogeneous. This is because, in heterogeneous agents, the service sets of the agents will be distinct from one another and therefore an efficient structure will need to connect every agent with all those agents providing services relevant to it and, at the same time, also help in equitable load distribution so that tasks are completed as quickly as possible. In contrast, for homogeneous agents, load balancing is the only feature that can be influenced by the structure because agents are all similar to each other in terms of the services that they can provide. To this end, 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. It is important to note that *SP* pertains to the complete service set *S* instead of having different values for individual services. This is because raising or decreasing the distribution of only some services will make the resultant trends unclear as we will not be aware of which services (those whose *SP* has been changed or those whose *SP* hasn't been changed) are more in demand by the tasks and hence cannot narrow down the performance to the difference in *SP* alone.

In static organisations, the service sets of the agents are unchanging across a simulation run. Hence, in our experiments for static organisations, we vary *SP* from 0 to 0.5 only (since we

TABLE 5.1: Values of the control variables

Parameter	Symb	Value assigned	Justification
Number of permanent agents	$\|A\|$	chosen from a uniform distribution between 4 – 20	Organisations generally have at least 4 members. Though the results shown here contain a maximum of 20 permanent agents, we have conducted experiments with larger numbers of agents (upto 100) and found broadly similar trends as shown here (see Section A.3 for these results).
Initial organisation structure	G	randomly generated	Since we experiment with thousands of runs and hundreds of tasks in each run, the initial structure will not carry much significance (see Section A.1).
Number of services	$\|S\|$	equal to $\|A\|$	The number of services possible is equal to the number of agents as the number of services matter only in terms of their distribution across the agents.
Computational rate of a service instance	p_i	chosen from a uniform distribution between 1 – 10	This values determines the other values like capacity, so we decided to give it a standard range.
Capacity of an agent	L_x	chosen from a uniform distribution between 11 – 20	Minimum is set to 11 so that the maximum computational rate is always less than the capacity. Maximum is set at 20 so that agents' capacities are filled up occasionally with 2 or 3 service instances.
Arrival agent of a task	a_x	randomly chosen from the set of agents A	The tasks enter the organisation, randomly, at any agent
Number of service instances per task	$\|SI_w\|$	chosen from a uniform distribution between 1 – 25	An upper limit of 25 is set so that the tasks do not become too bulky.
Arrival time of a task		chosen from a uniform distribution between 1 and end time of simulation	Since the tasks already have several other parameters that are varied, like NoP and pattern sets, the rate of arrival is maintained uniformly.
Communication cost coefficient	C	0.25	C is fixed at this value so that costs remain lower than the rewards obtained.
Management load coefficient	M	0.50	So that management causes more load than message passing but less than actual service instance execution
Reorganisation cost coefficient	D	1	So that the adaptation process is neither too cheap but nor is it hindered too much by the cost of reorganisation.
Reorganisation load coefficient	R	0.25	With this value, the adaptation deliberation will consume reasonable amounts of the capacity and require good meta-reasoning
Maximum pattern size		$\frac{1}{3}\|SI_w\|$	In the experiments where patterns are used for generating similar tasks, it is set to one-third of the maximum number of service instances in a task so that, in general, tasks need around three patterns.

verified that beyond 0.5, when the agents are quite homogeneous, the structures did not influence the performance significantly as the problem reduces to purely a load-balancing problem). However, in dynamic organisations, 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:

1. *SP* increases from 0 to 0.25 at a uniform rate.
2. *SP* decreases from 0.25 to 0 at a uniform rate.
3. *SP* increases at a uniform rate from 0 to 0.25 and then decreases back to 0.
4. *SP* decreases at a uniform rate from 0.25 to 0 and then increases back to 0.25.
5. *SP* changes suddenly from 0 to 0.25 midway through the simulation.
6. *SP* changes suddenly from 0.25 to 0 midway through the simulation.

We only limited *SP* to 0.25 for such dynamic scenarios because otherwise the rate of change becomes too high for the given limited simulation time duration and the distinction between the various scenarios blurs. Also, we find these six scenarios to be comprehensive enough as together they cover the general variations possible in a system, in this context.

5.1.2.2 Similarity between tasks

The other simulation parameter of importance is the kind of tasks entering the system. The tasks presented to the organisation 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 organisation structure should be able to adapt to the recurring task structures, thereby increasing the efficiency of the organisation. For example, if many tasks contain a dependency linking two services, then an efficient structure will tend to contain superior-subordinate or at least peer relations between the agents providing those two services. 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). Figure 5.1

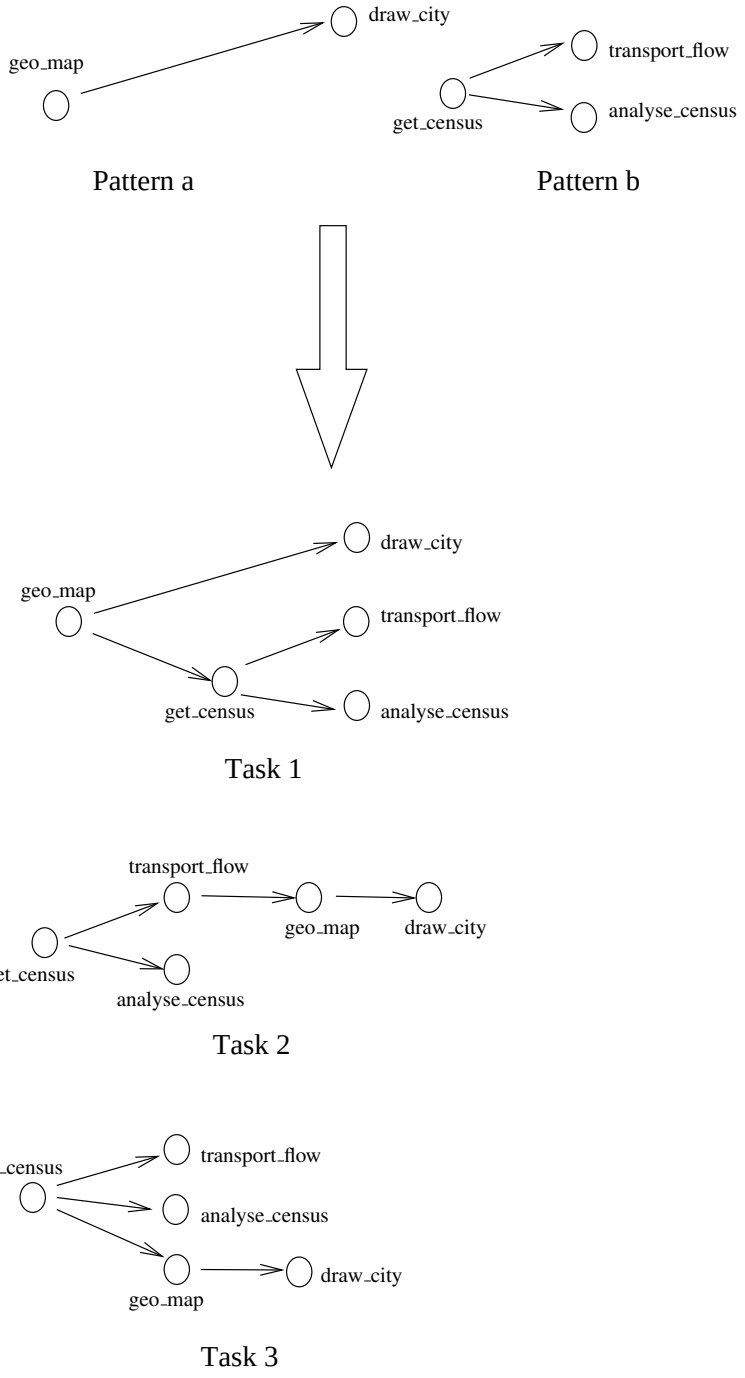


FIGURE 5.1: Patterns composing tasks

shows an example of how 2 patterns can be combined to form at least 3 different tasks. Thus, this method of generation enables us to control the similarity between the tasks using the number of patterns (NoP) as the parameter. Broadly speaking, tasks can be either similar or dissimilar. Hence, in our experiments, we mainly consider two scenarios:

1. completely dissimilar tasks ($NoP = \infty$).
2. highly similar tasks ($NoP = 5$).

In addition to these, we also varied the set of patterns being used within a simulation run to represent changing characteristics of the task environment. In more detail, within a simulation over a task environment containing similar tasks composed of patterns, even these frequently occurring patterns might change over time. That is, tasks coming in at a later time might be composed of patterns which are different from those composing the tasks at an earlier time. This change in the pattern set can again be either gradual or sudden. Therefore, we conducted a set of experiments for these varying task environments by changing the set of patterns within a simulation run in the following ways:

1. $NoP = 5$ chosen from 10 patterns in total; The pattern set contains 5 patterns at the outset. Each of these initial patterns is replaced with a new one at regular intervals such that towards the end of the simulation, all the 5 patterns are different from the ones at the beginning. Therefore, the pattern set is changed gradually.
2. $NoP = 2$ chosen from 10 patterns in total; same as before but only two patterns are used at a time in the pattern set. Hence, they are replaced multiple times and at a faster rate than before.
3. $NoP = 5$ chosen from 15 patterns in total; all the 5 patterns are replaced by 5 new ones suddenly at $\frac{1}{3}$rd of the total simulation time and the process is repeated again at $\frac{2}{3}$rds of the total time.

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. Also, static organisations face 1500 tasks (that is $|W| = 1500$) over 2000 time-steps to constitute one simulation run, while dynamic organisations face 3000 tasks ($|W| = 3000$) over 4000 time-steps for one simulation run. We provided more simulation time for dynamic organisations so that it is sufficient for the changes in the organisation (like changing service sets or task pattern sets) to take place. Finally, the set of agents A, is kept constant for closed organisations, while for the open ones, a randomly chosen number (up to a maximum of $|A|$) of temporary agents are added, as described in Section 3.2.4. The results shown here

are of experiments where the start-times and life-times 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 to be same as these (see Section A.2 for these results).

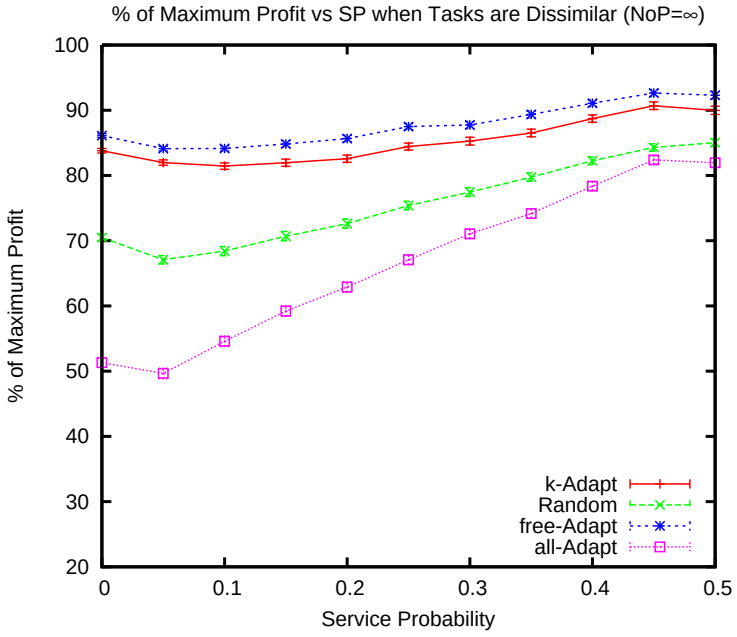
5.2 Results

We present the results in terms of the percentage of the maximum profit that is obtained by the organisation (averaged over the 1000 simulation runs as described above). The maximum profit is the profit obtained by `Central` which represents the theoretical upper bound. For static organisations, 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 organisations, *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*. A similar format is used even for scenarios with varying task environments.

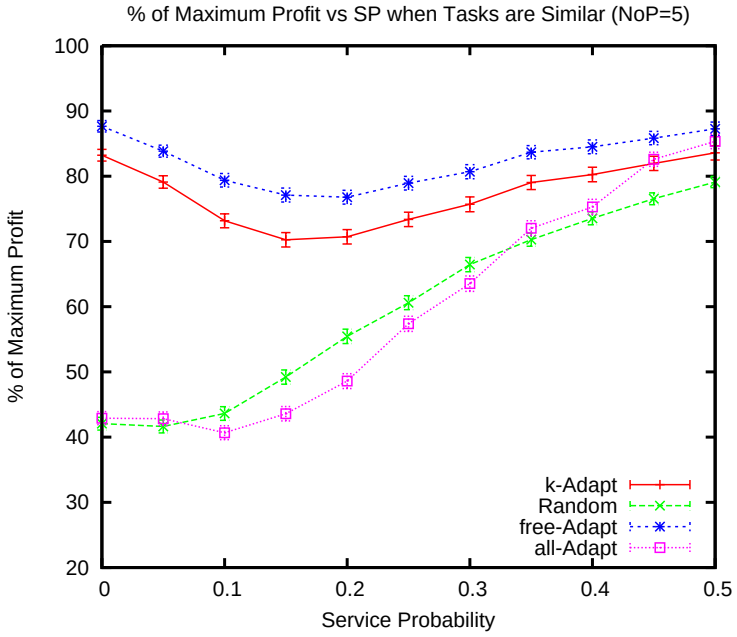
First we discuss the results of the experiments on static closed organisations (Section 5.2.1) and then for static open organisations (Section 5.2.2). Later on, we move onto dynamic closed (Section 5.2.3) and dynamic open organisations (Section 5.2.4). The final set of results refer to static and dynamic organisations placed in varying task environments with changing task pattern sets over time (Section 5.2.5).

5.2.1 Static Closed Organisations

Observing the results for static closed organisations, we find that in both the scenarios with dissimilar (Figure 5.2(a)) and similar tasks (Figure 5.2(b)), `k-Adapt` performs consistently better than `Random` . The difference in their performance narrows down (from the highest of 40% of profit to 10%) as the similarity of the agents increases because a smart method is correspondingly less useful when all the agents are homogeneous, as the significance of the structure itself diminishes. 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 organisations 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` improves with increasing *SP*. Also, studying the performance of `all-Adapt` , we see that in the case of similar tasks (Figure 5.2(b)), it performs better than `Random` when *SP* is 0. In this particular case of having similar tasks and unique agents, adaptation is most effective (as evident from the wide gap between `k-Adapt` and `Random`) and even



(a) Dissimilar tasks ($NoP = \infty$)



(b) Similar tasks ($NoP = 5$)

FIGURE 5.2: Average organisation profit for static closed organisations as *SP* increases

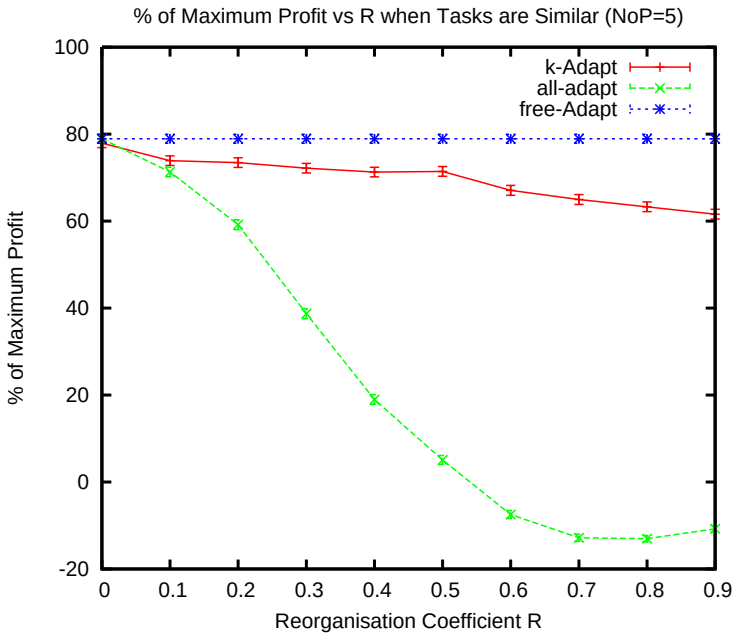


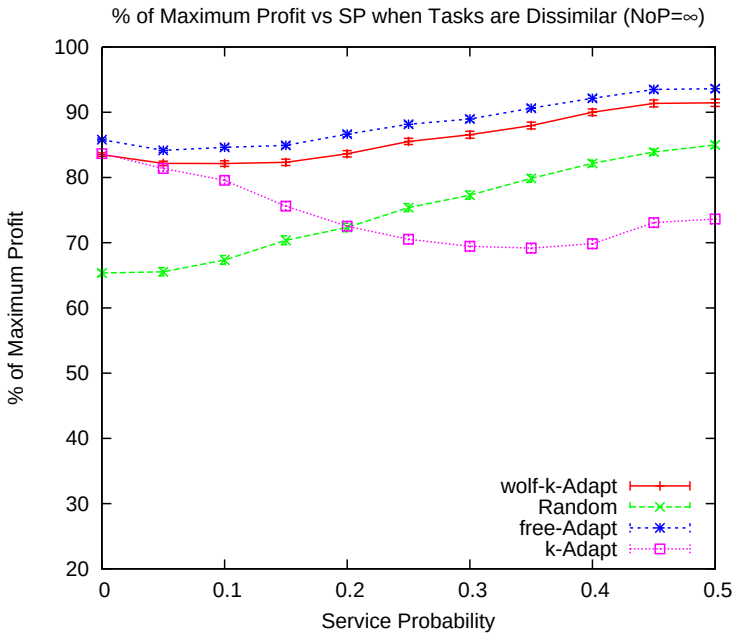
FIGURE 5.3: Average organisation profit for static closed organisations as R increases

`all-Adapt` is able to overcome its excessive usage of capacities for adaptation deliberation and perform better than `Random` . Similarly, `all-Adapt` 's performance improves greatly for higher values of SP $(0.35 - 0.5)$ because the resultant increase in the average capacity per service (as explained above) helps `all-Adapt` to use up the capacity for the adaptation reasoning without having to delay the tasks very much. This is also the reason that it manages to perform slightly better than `k-Adapt` when SP is 0.5 for similar tasks.

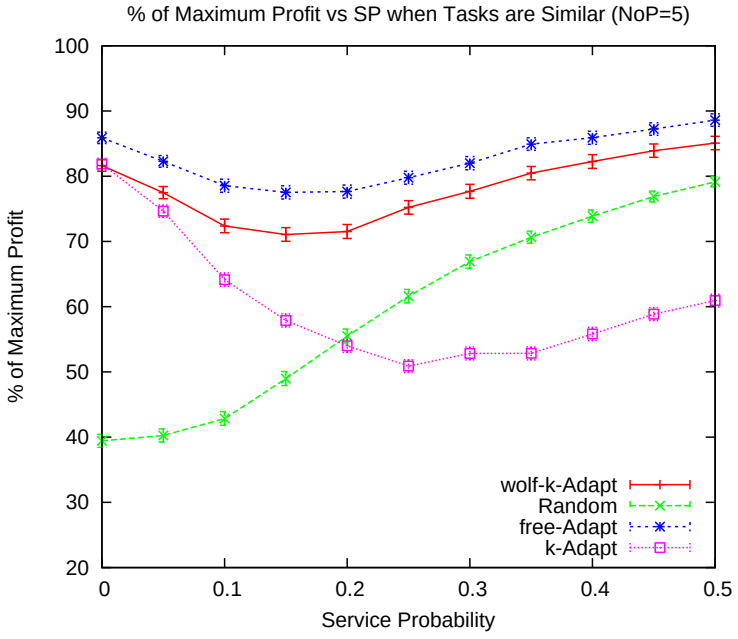
We also conducted experiments by varying R from 0 to 0.9 (see Figure 5.3) to focus on the effectiveness of the meta-reasoning aspect in the adaptation process. We found that the fall in the performance of `k-Adapt` is gradual and minimal, while it is drastic in `all-Adapt` (though, the result shown here is for similar tasks, the same trend is seen for dissimilar tasks as well). 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 meta-reasoning is a crucial aspect in an adaptation process and cannot be ignored. Moreover, we see that the performance of `k-Adapt` is always close to that of `free-Adapt` , thus confirming the efficacy of our meta-reasoning approach.

5.2.2 Static Open Organisations

In the case of static open organisations, we find that `wolf-k-Adapt` performs considerably better than `Random` when tasks are both dissimilar (Figure 5.4(a)) and similar (Figure 5.4(b)), thus confirming that the adaptation method (including the WoLF principle) is helping organisations to maintain the performance even when agents are added and removed. More importantly, `k-Adapt` , which does not make use of the WoLF principle, degrades rapidly as the similarity between agents increases. In fact, as SP increases beyond 0.2, it's performance is worse than



(a) Dissimilar tasks ($NoP = \infty$)



(b) Similar tasks ($NoP = 5$)

FIGURE 5.4: Average organisation profit for static open organisations as *SP* increases

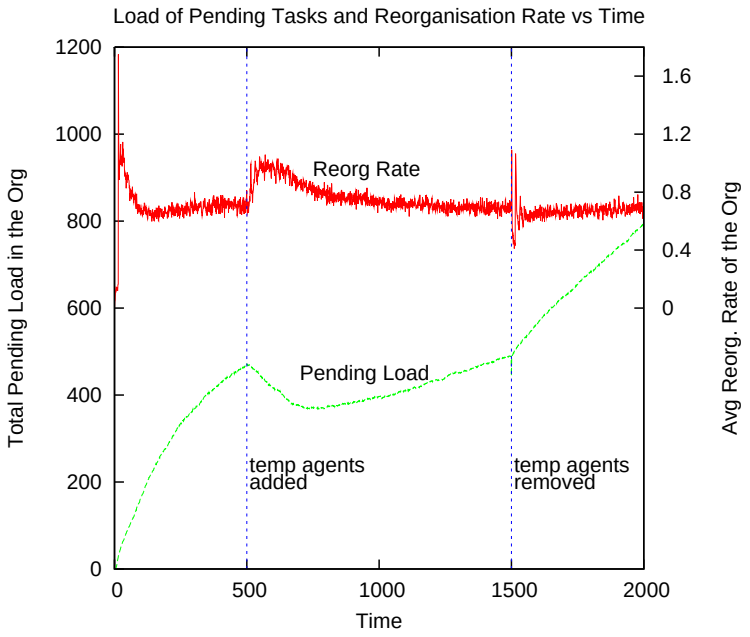


FIGURE 5.5: Showing changes to pending load and reorganisation rate when agents are added and removed

that of `Random` . This shows that, without the WoLF principle, `k-Adapt` is unable to properly assimilate and use the new agents entering the system. This inability does not affect the performance very much when agents are mostly unique (*SP* is close to 0), because the new agents will rarely be providing the services that are mainly in demand in the organisation. However, with increasing *SP* as most of the new agents are capable of most of the services, distributing the load onto the new agents becomes imperative and `k-Adapt` is unable to effect it. In these scenarios, even `Random` is able to perform better than `k-Adapt` because the new agents are somewhat assimilated and allocated tasks through the randomised changes happening to the structure. However, in all these scenarios, `wolf-k-Adapt` performs much better than either of these two methods showing that the WoLF principle is very useful for assimilating new agents into the organisation and maintaining the performance.

Furthermore, Figure 5.5 gives us an insight into what is happening to the organisation 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 organisation (left y-axis) across the time duration of the simulation, and shows the corresponding reorganisation rate in terms of the number of relations in the organisation changed in a time-step (right y-axis). For these experiments, we fixed $SP = 0.20$ and $NoP = \infty$. We observe a gradual fall in the load starting at time=500 corresponding to when temporary agents are added. Also at 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 reorganisation rate, we find that it is high in the beginning and then

TABLE 5.2: Profit for dynamic closed organisations with dissimilar tasks ($NoP = \infty$)

SP variance	wolf-decay-Adapt	wolf-k-Adapt	Random
$0 \rightarrow 0.25$ (gradually)	93.55%($\pm$0.30)	88.65%($\pm$0.35)	76.23%($\pm$0.46)
$0.25 \rightarrow 0$ (gradually)	88.26%($\pm$0.40)	77.93%($\pm$0.45)	72.42%($\pm$0.56)
$0 \rightarrow 0.25 \rightarrow 0$ (gradually)	92.13%($\pm$0.33)	86.52%($\pm$0.37)	74.85%($\pm$0.49)
$0.25 \rightarrow 0 \rightarrow 0.25$ gradually)	90.43%($\pm$0.33)	80.96%($\pm$0.39)	74.68%($\pm$0.49)
$0 \rightarrow 0.25$ (suddenly at $t = 2000$)	91.85%($\pm$0.32)	87.34%($\pm$0.37)	74.27%($\pm$0.48)
$0.25 \rightarrow 0$ (suddenly at $t = 2000$)	90.72%($\pm$0.32)	80.44%($\pm$0.40)	74.80%($\pm$0.49)

TABLE 5.3: Profit for dynamic closed organisations with similar tasks ($NoP = 5$)

SP variance	wolf-decay-Adapt	wolf-k-Adapt	Random
$0 \rightarrow 0.25$ (gradually)	74.78%($\pm$1.01)	69.99%($\pm$1.07)	55.72%($\pm$1.09)
$0.25 \rightarrow 0$ (gradually)	78.80%($\pm$1.01)	67.73%($\pm$1.04)	46.99%($\pm$0.99)
$0 \rightarrow 0.25 \rightarrow 0$ (gradually)	76.52%($\pm$1.04)	70.81%($\pm$1.05)	53.71%($\pm$1.03)
$0.25 \rightarrow 0 \rightarrow 0.25$ gradually)	76.27%($\pm$0.99)	66.36%($\pm$1.03)	50.64%($\pm$1.05)
$0 \rightarrow 0.25$ (suddenly at $t = 2000$)	76.32%($\pm$1.02)	71.38%($\pm$1.02)	51.31%($\pm$1.08)
$0.25 \rightarrow 0$ (suddenly at $t = 2000$)	82.11%($\pm$0.97)	71.15%($\pm$0.98)	50.15%($\pm$0.96)

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.

5.2.3 Dynamic Closed Organisations

We see that for dynamic closed organisations, wolf-decay-Adapt consistently performs significantly better than both wolf-k-Adapt and Random in both the scenarios with dissimilar (Table 5.2) and similar (Table 5.3) tasks. It is notable that for dissimilar tasks, wolf-decay-Adapt is able to reach 90% of the maximum profit which is 5-10% and 15% better than wolf-k-Adapt and Random respectively. In this context, we also observe that the performance of wolf-k-Adapt when the service probability *SP* (introduced in Section 5.1.2) is reduced is worse than when *SP* is increased. This is because in organisations 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. Yet, 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 (allocations of services instances containing a service to an agent which previously did not provide that service) will also be possible which wolf-decay-Adapt is capable of identifying much more quickly than wolf-k-Adapt because it gives more weight to most recent interactions.

TABLE 5.4: Profit for dynamic open organisations with dissimilar tasks ($NoP = \infty$)

SP variance	wolf-decay-Adapt	wolf-k-Adapt	decay-Adapt	Random
$0 \rightarrow 0.25$ (gradually)	92.85%($\pm$0.33)	87.90%($\pm$0.38)	79.05%($\pm$0.36)	72.63%(0.51)
$0.25 \rightarrow 0$ (gradually)	87.71%($\pm$0.42)	77.04%($\pm$0.48)	78.87%($\pm$0.44)	68.95%(0.61)

TABLE 5.5: Profit for dynamic open organisations with similar tasks ($NoP = 5$)

SP variance	wolf-decay-Adapt	wolf-k-Adapt	decay-Adapt	Random
$0 \rightarrow 0.25$ (gradually)	75.41%($\pm$1.08)	70.85%($\pm$1.10)	61.30%($\pm$0.80)	50.00%(1.09)
$0.25 \rightarrow 0$ (gradually)	78.84%($\pm$1.02)	67.77%($\pm$1.05)	65.60%($\pm$0.99)	40.98%(0.99)

In this 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 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. This shows that 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.

5.2.4 Dynamic Open Organisations

For dynamic open organisations, we observe that wolf-decay-Adapt significantly outperforms the other methods in all cases for both dissimilar (Table 5.4) and similar (Table 5.5) tasks. In more detail, wolf-decay-Adapt is better than wolf-k-Adapt by 5% to 10% depending on whether *SP* is increasing or decreasing. This variance in the performance of wolf-k-Adapt can be explained in the same way as in dynamic closed organisations (Section 5.2.3) discussed above. Moreover, wolf-decay-Adapt is consistently 10-14% better than decay-Adapt and and 20-35% better than Random . This shows that both the extensions detailed in Section 4.2 are critical for a good performance in these organisations with agents moving in and out and changing properties.

TABLE 5.6: Profit for organisations facing tasks with $NoP = 5$ out of a total 10 patterns, changed gradually, one at a time

SP variance	wolf-decay-Adapt	wolf-k-Adapt	Random
0 (no variance)	87.11%($\pm$0.82)	73.75%($\pm$0.88)	40.36($\pm$0.92)
0.25 (no variance)	70.20%($\pm$1.12)	63.08%($\pm$1.14)	52.90($\pm$1.11)
$0 \rightarrow 0.25$ (gradually)	76.36%($\pm$0.99)	69.76%($\pm$1.00)	57.61%($\pm$1.00)
$0.25 \rightarrow 0$ (gradually)	79.36%($\pm$0.95)	66.87%($\pm$0.98)	48.32%($\pm$0.92)

TABLE 5.7: Profit for organisations facing tasks with $NoP = 2$ out of a total 10 patterns, changed gradually, one at a time

SP variance	wolf-decay-Adapt	wolf-k-Adapt	Random
0 (no variance)	83.82%($\pm$0.77)	65.15%($\pm$0.78)	34.08($\pm$0.69)
0.25 (no variance)	61.83%($\pm$1.12)	54.00%($\pm$1.12)	47.08($\pm$1.00)
$0 \rightarrow 0.25$ (gradually)	66.78%($\pm$0.98)	56.04%($\pm$0.98)	43.76%($\pm$0.88)
$0.25 \rightarrow 0$ (gradually)	74.17%($\pm$0.86)	58.62%($\pm$0.87)	39.37%($\pm$0.73)

TABLE 5.8: Profit for organisations facing tasks with $NoP = 5$ out of a total 15 patterns, changed suddenly in sets of 5

SP variance	wolf-decay-Adapt	wolf-k-Adapt	Random
0 (no variance)	85.58%($\pm$0.80)	70.37%($\pm$0.85)	43.44($\pm$0.90)
0.25 (no variance)	69.97%($\pm$1.10)	62.20%($\pm$1.11)	54.26($\pm$1.11)
$0 \rightarrow 0.25$ (gradually)	76.44%($\pm$0.97)	69.01%($\pm$0.98)	57.76%($\pm$1.00)
$0.25 \rightarrow 0$ (gradually)	78.28%($\pm$0.95)	64.96%($\pm$0.94)	50.31%($\pm$0.91)

5.2.5 Varying Task Environments

Finally, we look at the results of experiments conducted for varying task environments where the pattern set composing the tasks is changed with time during a simulation (Tables 5.6, 5.7 and 5.8). Again we see that wolf-decay-Adapt performs significantly better than wolf-k-Adapt and Random in all the cases and scenarios. The efficacy of the decay approach in such task environments can be clearly seen in the first two cases where *SP* is kept constant in all the three scenarios. Here, we find that wolf-decay-Adapt performs 7 to 15% better than wolf-k-Adapt . Moreover, by comparing the third and fourth cases of Table 5.6 with the first two cases of dynamic closed organisations (Table 5.3) which have the same *SP* variance, we observe that when patterns are changing, the gap between the performance of wolf-k-Adapt and wolf-decay-Adapt increases, while that between wolf-k-Adapt and Random reduces. This strengthens our claim that the decay approach is useful to maintain performance in such changing task environments.

Furthermore, we see that the trends in the scenario when the pattern set contains only 2 patterns at a time (Table 5.7) are similar to when it contains 5 (Table 5.6), thus assuring that the improvement of the performance of wolf-decay-Adapt over wolf-k-Adapt is not dependent on the degree of similarity of the tasks. In a similar vein, we also see that the respective values in the scenario when the pattern set is changed suddenly (Table 5.8) are very close to the scenario when they are changed gradually (Table 5.6). This mirrors the same trend observed for the dynamic organisations where it did not matter whether *SP* was changed gradually or suddenly in

the middle of the simulation. Thus, these results tell us that the performance of the system is not affected by the gradient of the dynamism as long as the total change averaged over the total time period is the same.

In summary, we find that, on average over all the settings, our adaptation method performs at 80% of the omniscient centralised allocation method. Furthermore, on average, it is 20% better than a random reorganisation approach (reaching up to a maximum of 45%). The results for open and dynamic organisations show that the respective enhancements, the WoLF principle and the decaying weights, are crucial for maintaining the performance. In particular, we see that on average over the settings, the performance of the method with the respective enhancement is 8% better than without it.

5.3 Summary

In this chapter, we first presented the experimental setup used for evaluating our structural adaptation mechanism by listing the other adaptation and allocation methods for comparison and then describing the various simulation parameters. Particularly, service probability *SP* is used to determine the degree of homogeneity of the agents, and the number of patterns *NoP* determine the similarity of the tasks coming into an organisation. From our experiments, we found that our mechanism always performed better than a randomly reorganising approach. Furthermore, its performance is comparable to the organisations in which the agents use an omniscient central repository to perform shortest and best possible allocations without using any resources. Therefore, it is evident that our method is successful in improving the performance of the organisation and brings its performance closer to that of a centralised infinitely resourceful allocator than a randomly adapting organisation. Furthermore, we see that for open organisations, our enhancement to the adaptation method, using the WoLF principle, performs significantly better than without it. Similarly, for dynamic organisations, using the decay mechanism in the adaptation process is better at helping in maintaining the performance than without using it, as the properties of the agents change over time. In view of these results, we claim that our adaptation mechanism satisfies the requirement of developing a decentralised agent-based structural adaptation method for improving the performance of problem-solving organisations. Hence, we have contributed to advancing the state of the art in the domain of adaptation in agent organisations as we mentioned in Section 1.4.

Chapter 6

Conclusions and Future Work

This chapter concludes the thesis by first summarising the research and matching it up to the objectives and contributions to the state of the art that were laid out in Chapter 1. Following that, we discuss the different ways of extending this research in the future.

6.1 Summary

As stated in Chapter 1, the focus of this thesis is to aid in the advancement of autonomic systems by developing decentralised structural adaptation mechanisms for problem-solving agent organisations based on the paradigm of self-organisation. As noted in Section 1.3, we divided this overarching research objective into three parts:

1. An agent organisation framework that serves as an abstract representation of distributed computing systems.
2. A self-organisation based decentralised structural adaptation method that the agents can use in an organisation to improve the performance of the overall organisation, especially in resource-constrained dynamic environments.
3. An empirical evaluation mechanism that tests the efficacy of such an adaptation method.

In the following, we summarise this thesis by looking at our contributions towards meeting each of these requirements.

In Chapter 3, we introduced an abstract agent organisation framework for depicting distributed computing systems. We presented our model by detailing our representation of the task environment and the organisation along with a performance evaluation system. Specifically, the tasks are made up of service instances, each of which specifies the particular service and the computation required. The organisation consists of agents providing services and having computational capacities. The structure of the organisation manifests the relationships between the

agents and regulates their interactions. Any two agents in the organisation could be strangers, acquaintances, peers or superior-subordinates. The relations of the agents determine what service information is held by the agents about the other agents and how to allocate service instances to them. We also presented the coefficients that affect the environment (communication cost, management load, reorganisation load) and the functions for calculating the organisation's cost and reward, thus enabling us to evaluate the profit obtained by it when placed in a dynamic task environment. Hence, our organisation framework provides a simulation platform that can be used by designers to implement and test their adaptation techniques before porting them to real and domain-specific systems. In particular, we designed our model such that the agents, though generic, realistically represent the components that would compose autonomic systems. The organisation is decentralised and agents possess local views and limited capacities like any large distributed computing system. Nevertheless, the agents interact with each other based on the organisation structure, which also influences the task allocations and, thereby, the organisational performance. In this context, our framework provides sufficient flexibility for the agents to modify their characteristics and social interactions, that is, manage themselves, just as expected in autonomic systems. Furthermore, we also provided the reorganisation cost (D) and load coefficients (R) to represent the price of adaptation. Thus, we have presented an organisation framework that can be used as a platform for developing adaptation techniques, especially focusing on the agents' social interactions, thereby satisfying the requirements stated in Section 1.3.1. As a result, this chapter has helped us achieve the research contribution mentioned in Section 1.4.

Next, in Chapter 4, we presented a structural adaptation method that can be applied individually and locally by all the agents in order to improve the organisation's performance. Using our method, a pair of agents jointly calculate the utility of changing their inter-relation and take the appropriate action accordingly. They do this through a value function whose attributes denote the possible increase or decrease to the load on the agents and communication costs when the action is taken. These attribute values are estimated by the two agents on the basis of their history of interactions. Moreover, our method also enables an agent to meta-reason about when and with whom to initiate this adaptation deliberation. Specifically, an agent varies the number of other agents that it initiates this deliberation with, in a time-step, depending on the available capacity at itself and the rate of adaptation occurring in the immediate past. Additionally, we extend our method so that it performs well even in open organisations 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 enhanced our method to tackle dynamic organisations 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.

It is evident that our adaptation method works purely by redirecting agent interactions, thereby, changing the organisation 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 in the organisations 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 organisation through adaptation. Also, the method does not require any feedback from the execution of the tasks and, as a result, is applicable for a wide range of task environments. On this note, it is also clear that, being based on the agent interactions, the method is not tied to the task allocation mechanism used by the agents and can be similarly used in systems wherein agents use a different allocation procedure. Finally, since the adaptation method is purely agent-based, decentralised and continuous over time, it satisfies the principles of self-organisation (discussed in Section 2.2) which require that the method be autonomous without any external control, be continuous and need no central authority. By so doing, we have succeeded in meeting the requirements listed in Section 1.3.2 which aim at an adaptation method for agent organisations that is decentralised, continuous and based on local reorganisation towards causing benefit to the organisation as a whole. Moreover, we have also satisfied the requirements regarding the meta-reasoning involved in aiding the agents to decide when to initiate adaptation reasoning and also developed the method such that it continues to perform well even in open and dynamic organisations where the agents and their properties are changing with time. Through achieving all these requirements, we have succeeded in advancing the state of the art in terms of the research contributions listed in Section 1.4. Having been developed on an abstract organisation platform, the method is generic and applicable to any cooperative agent organisation requiring sustained inter-agent interactions for achieving task objectives in a resource-constrained environment.

With our adaptation method in place, we sought to demonstrate its effectiveness in Chapter 5. In particular, we evaluated our approach empirically by varying interesting simulation parameters like the heterogeneity of the agents and the similarity of the tasks, in addition, to the openness and dynamism of the organisations. The heterogeneity of agents is represented through the distribution of service capabilities across the agents in the organisation, while the similarity between tasks is modelled by composing the tasks from stereotypical task components called patterns. We found that our method performs at 80% (average over all settings) of a centralised omniscient allocation method which is 20% (average over all settings) better than a randomly adapting method. Both the relevant enhancements to our basic method, WoLF principle and decaying weights, are seen to be useful for maintaining the good performance in the face of open and dynamic organisations. In addition, the decay approach is also seen to perform well for varying task environments in which the kind of similarity that might exist across tasks changes with time. With these results, we see that a smart adaptation method is needed for a considerably better performance of organisations placed in dynamic environments when compared with a random approach. Furthermore, organisations using such a decentralised adaptation method are able to deliver performance close to those using an unrealistic centralised, all-knowing,

infinitely resourceful allocator, while also simultaneously maintaining the robustness of the system. By demonstrating the efficacy of the adaptation method over different parameters, this chapter satisfies the third and final set of requirements for this thesis listed in Section 1.3.3.

During the experiments, we also observed that the time taken for the simulations of organisations using our adaptation method was approximately equal to that taken by randomly adapting organisations. This suggests that our adaptation method is not computational intensive and uses only as much resources as a randomly adapting method does. Moreover, in general, the computational amounts needed by an adaptation method (determined by the reorganisation load coefficient R in our framework) will be much less than the computational amounts required for processing the tasks (p_i) in real systems.

The characteristics of our adaptation method, discussed earlier, make it suitable to 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 paper demonstrates a practical and robust, decentralised approach for continuous self-adaptation of problem-solving agent organisations, thereby providing an important component for the development of autonomic systems. Specifically, our adaptation method can be incorporated into the various components of an autonomic system to help in improving the performance of the system, as we envisaged in Chapter 1. Using the method, all the components will be able to self-manage their interactions with the other components in the system to optimise the performance of the system and also aid it in facing any changes that might be taking place, both internally in the system or in the external environment. Particularly, any computing system which has the following characteristics may find our method beneficial:

- A distributed system with no central authority or access to global-level knowledge.
- Containing cooperative and autonomous components working solely towards system-wide goals.
- Placed in unknown task environments with no prior information about the tasks.
- Where task-solving requires the components to interact by collaborating with each other for allocation and execution of the parts of the tasks.
- Where interactions between the components are not cheap. The interactions consume resources and require regulation based on a structure.

By developing the method, we have succeeded in the advancing the state of the art in the domain of adaptation of agent organisations.

6.2 Future Work

Though we have developed an effective adaptation method for agent organisations, there are still a number of avenues for advancing the work presented in this thesis. In view of this, we identify the different branches for possible extensions by revisiting each of the three primary requirements (listed in Section 1.3) that we sought to achieve in this thesis, before broadening out into the realms of autonomic computing in general.

First, looking at our agent organisation model, we have represented the resources at the agents in terms of the computational capacities being used for providing services. However, there are other aspects to resources like memory or network bandwidth. Though we have modelled network resources through the communication costs, we have not limited it in the way we did for the computational capacity of the agents. Therefore, future work is needed to extend the organisation framework to consider such limited resources like memory and network in addition to the current ones. This will not only make the framework more representative of the real systems but also bring forth newer challenges for the adaptation method, like having to optimise over different kinds of resources at the same time and trading off one for the other depending on the conditions. Another stream of work in this regard can involve modelling failures in task execution and reallocation of such failed service instances. Failures in execution are a real-life occurrence and methods towards handling and re-provisioning of those tasks will add to the robustness of the system.

The focus of the second branch of future work is on the adaptation method itself. Right now, the 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 organisation's agents. Hence, the method can be extended such that the agents can adapt in a *proactive* fashion when such information is available to them. For example, if the agents are aware that there will be a sudden influx of tasks containing dependencies with two particular kinds of services, the agents might consider it while adapting and relations might be formed between agents providing those two services before hand, in anticipation of those forthcoming tasks. More generally, these expectations of dynamism in the future might be modelled using probability distributions which can then be taken into account by the agents during adaptation. For example, the agents might model the expectant changes to the loads and costs in the organisation due to some predicted future changes in tasks or agents, weigh these on the basis of the probability of these changes actually taking place, and then use these values together with the current values in the utility functions. Such a future-sensitive method will be better at utilising all the information when available, and will also result in an organisation better prepared to deal with any predicted changes.

Another possible avenue for extending the adaptation method involves making it a multi-step process. Such an adaptation approach will involve three or more agents at the same time therefore resulting in multiple steps of restructuring. Involving a group of agents rather than just

two agents will provide them with a wider view of the system. More information, including the corresponding structure between the group of agents (like a sub-graph) and loads on the agents might possibly help them to adapt better. Doing this could involve some kind of planning and deliberation within the group so that they are able to look ahead and carry out a sequence of adaptation actions. However, there is a trade-off to such an approach as the adaptation process might lose some of the decentralisation. This is because the information from the agents in the group will have to be collated somewhere and the intra-group adaptation might need to be coordinated by a temporary leader. Yet, such a concerted adaptation approach will be useful in particular scenarios where complete decentralisation is not of a high priority and small improvements in performance are also highly regarded. Taking this line of thought further, such intra-adapting groups within an organisation can be considered to be *departments*. Therefore, in some cases, it might be useful to break down the adaptation process into intra-department and inter-department and so on. Recognising and delineating such departments for adaptation is also an interesting problem to be tackled. This might require the agents to identify which parts of the structure are performing poorly using only their limited information. Similarly, the agents might have to break up the structure into clusters with relatively low interaction between them. These challenges provide impetus towards adding aspects of graph theory into the adaptation mechanism. At the same time, this kind of group based adaptation will be useful in extremely large organisations where most agents interact with only a small subset of the agents and are mostly unaware of the remaining agents in the system.

The third way of extending the work presented in this thesis is through the evaluation mechanism. 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 could 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 the deployment in the real-life systems.

In the above, we have proposed several ways of extending the work presented in this thesis. However, considering the broader goals of this research, much more work is still needed before autonomic systems become an everyday reality. The self-management property of these systems also entails characteristics like self-healing and self-protection which require more context-aware solutions wherein the components explicitly evaluate the conditions and take appropriate measures. Self-management not only requires maintenance of the interactions within the system (as addressed in this thesis), but also internal reconfiguration of the individual components themselves. These challenges become even harder when these components are developed at different places using different technologies and by different developers. Indeed, the need for autonomic systems has also arisen, in part, due to the increasing openness and distributed nature of the

latest computing systems which are being based on service oriented architectures, in addition to the increasing complexity of the traditional distributed systems. Such issues further underline the importance of agent-based approaches for achieving the goals of autonomic computing. However, as rightly pointed out by Brazier et al. (2009), the ongoing development of autonomic systems still does not use the multi-agent paradigms or the agent development toolkits to the fullest extent. On the other hand, development of multi-agent systems also tends to be restricted to the theoretical domain rather than being incorporated into the existing applications. Therefore, active exchange of ideas and collaboration between these two fields is bound to unlock a vast potential for research that will, in turn, promote quicker and faster development of both the fields.

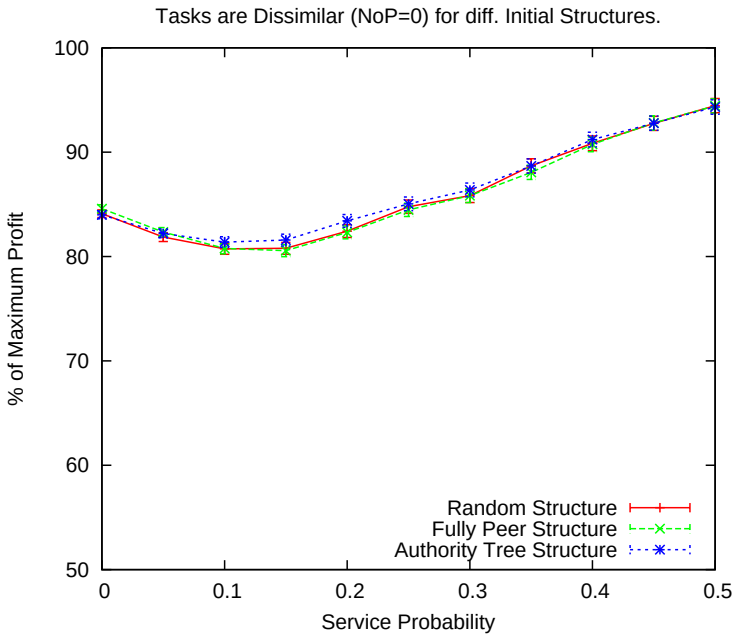
Appendix A

Additional Results

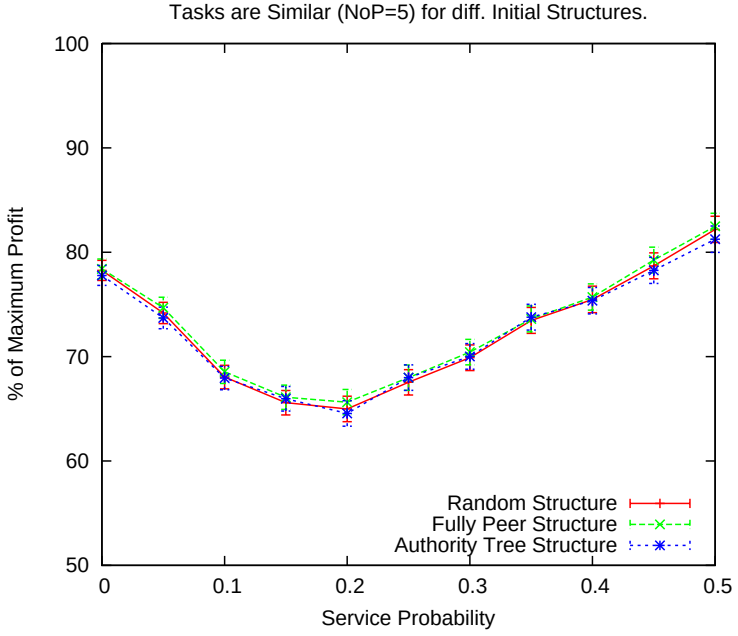
In this appendix, we present the set of additional results that have been mentioned in Chapter 5. In particular, the next section discusses experiments relating to the initial structure generated for the simulations. Following that, Section A.2 describes the results obtained by varying the start-times and life-times of the temporary agents in open organisations. Then, Section A.3 presents the results of experiments conducted over open organisations containing a maximum of 100 agents at a time. Finally, Section A.4, compares methods using linear and exponential decay functions for closed dynamic organisations.

A.1 Initial Structure of the Organisation

For all the experiments presented in Chapter 5, the initial structure of the organisation was generated randomly (as mentioned in Table 5.1). We asserted that the starting structure of the organisation will not carry much significance as it will be modified by the adaptation process. To prove this claim, we conducted experiments over closed static organisations by varying the initial structure. In particular, we considered organisations starting with a completely connected peer graph (Fully Peer Structure) and those with a complete hierarchical structure in the form of a tree of superior-subordinate relations (Authority Tree Structure) along with the generally used organisations consisting of a randomly generated initial structure (Random Structure). All of these three cases use the same adaptation method (`k-Adapt`). From the results, we find that for both dissimilar (Figure A.1(a)) and similar (Figure A.1(b)) tasks, there is no significant difference in the performance of the organisation. This confirms our assertion that the initial structure of the organisation does not play an important role in its performance when equipped with our adaptation method.



(a) Dissimilar tasks ($NoP = 0$)



(b) Similar tasks ($NoP = 5$)

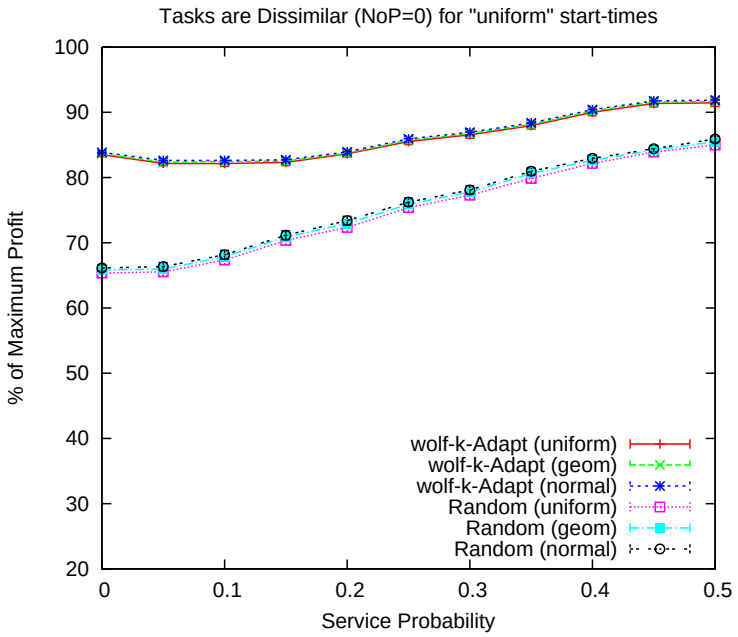
FIGURE A.1: Average organisation profit for static closed organisations with different initial structures

A.2 Distribution of Start-times and Life-times

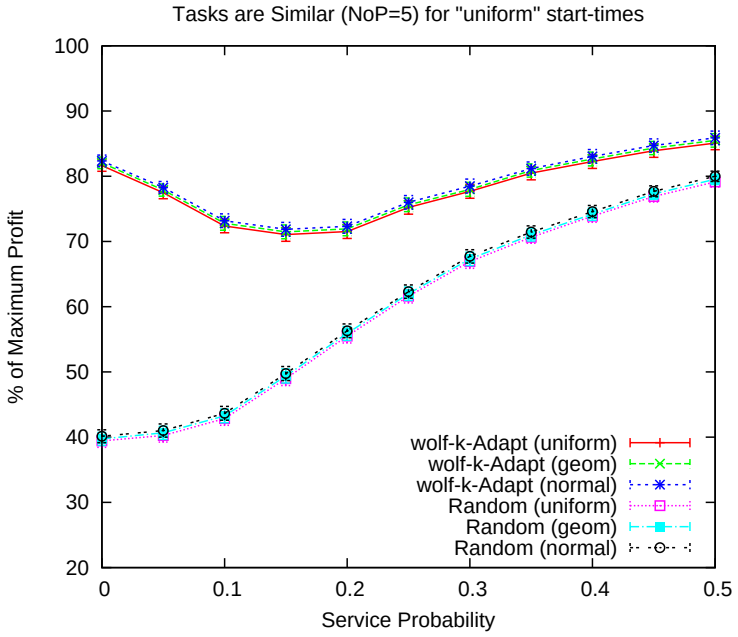
The results for static open organisations that were presented in Section 5.2.2, are based on settings where the start-times and life-times of the temporary agents are chosen from a uniform distribution. Here, we show the results with other distributions. In particular, Figure A.2 refers to the settings where the start-times are chosen from a uniform distribution for a combination of the distribution of the life-times. Similarly, Figure A.3 presents the set of results when the start-times are chosen from a normal distribution. In each of these, we present the profits obtained by the two methods, `wolf-k-Adapt` and `Random` , when the life-times of the temporary agents are chosen from uniform, normal and geometric distributions (as marked in the legend). In all of these results, we find that there is no discernible difference in the performance when the start-time or the life-time distribution is varied. This confirms our assertion that the distribution of the start-times or life-times of the temporary agents play no significant part in the performance of any of the methods.

A.3 Open Organisations with upto 100 agents

All the experiments discussed in Chapter 5 were based on the settings where the maximum number of agents in the organisation was limited to 20 permanent agents ($|A| <= 20$) and in the case of open organisations, the number of temporary agents additionally added was again limited to $|A|$. Therefore, the number of agents in the organisation never crossed 40. In this section, we present results of experiments conducted on open organisations, where the maximum number of the permanent and temporary agents is set at 50 each. Therefore, these represent scenarios wherein the number of agents in the organisation can reach upto 100. In particular, Figure A.4(a) and Figure A.4(b) present the results for dissimilar and similar tasks respectively. Studying these results, we observe the same trends as those seen in Section 5.2.2. However, the gap between the various methods has comparatively widened. That is, `wolf-k-Adapt` obtains a lower percentage of the maximum profit than in the earlier scenarios which contained lesser number of agents. This shows that efficient structural adaptation is relatively harder with a greater number of agents as it involves more entropy that needs to be countered by the method. However, it is heartening to note that the difference in performance between `free-Adapt` and `wolf-k-Adapt` remains the same as before, thus proving that our meta-reasoning approach performs equally well even when the number of agents is increased greatly. On the same lines, the performance of `Random` is much worse than before. In fact, we see that in the case of similar tasks (see Figure A.4(b)), it falls below 0 for heterogeneous agents (when *SP* is close to 0). This happens because `Random` finds it much more harder to link agents suitably well when there are more number of them in the system. Moreover, when the agents have unique service sets, randomly adapting the structure is much less likely to efficiently link useful agents together.

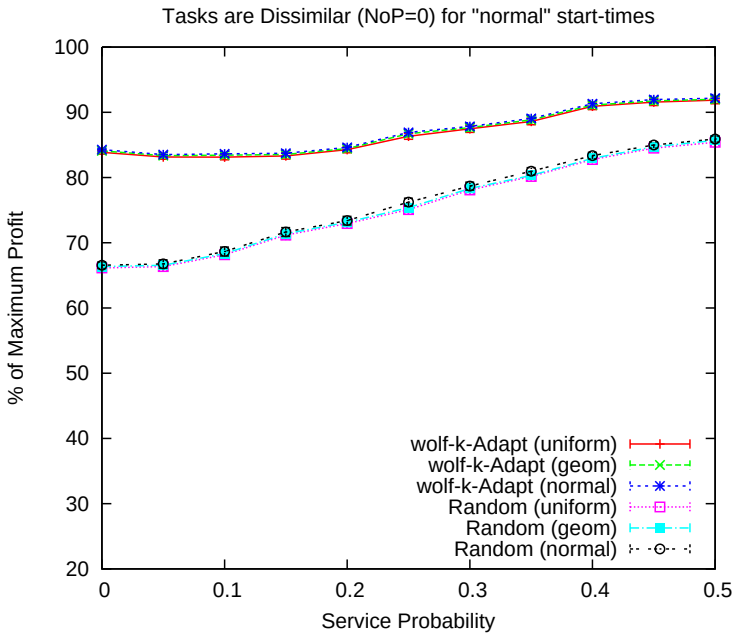


(a) Dissimilar tasks ($NoP = 0$)

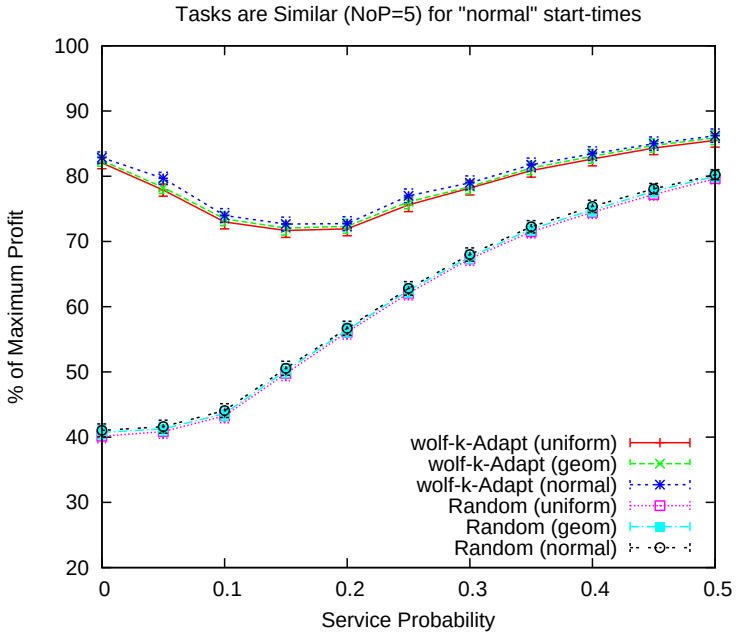


(b) Similar tasks ($NoP = 5$)

FIGURE A.2: Average organisation profit for static open organisations with start-times of the temporary agents chosen from uniform distribution

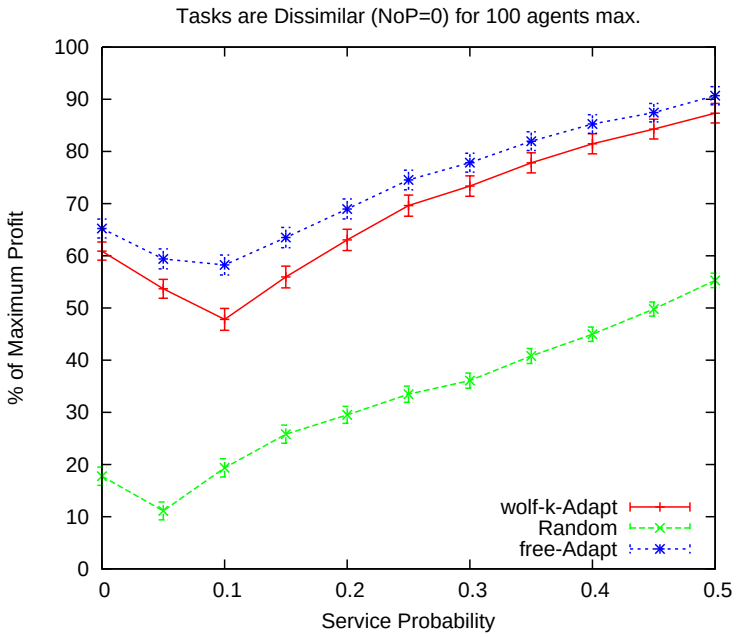


(a) Dissimilar tasks ($NoP = 0$)

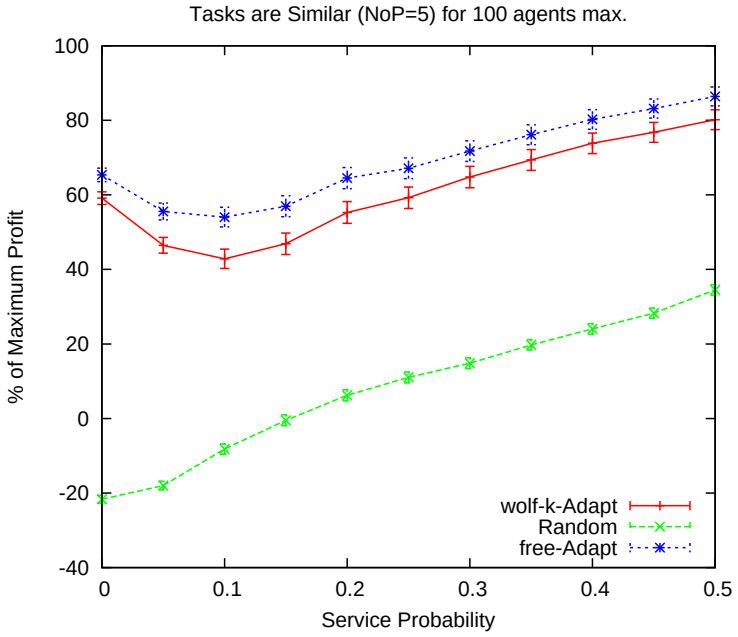


(b) Similar tasks ($NoP = 5$)

FIGURE A.3: Average organisation profit for static open organisations with start-times of the temporary agents chosen from normal distribution



(a) Dissimilar tasks ($NoP = 0$)



(b) Similar tasks ($NoP = 5$)

FIGURE A.4: Average organisation profit for static open organisations with a maximum of 100 agents

TABLE A.1: Profit for dynamic closed organisations with dissimilar tasks ($NoP = 0$)

SP variance	wolf-decay-Adapt (exponential)	wolf-decay-Adapt (linear)	wolf-k-Adapt
$0 \rightarrow 0.25$ (gradually)	92.06%($\pm$0.34)	93.55%($\pm$0.30)	88.65%($\pm$0.35)
$0.25 \rightarrow 0$ (gradually)	84.77%($\pm$0.44)	88.26%($\pm$0.40)	77.93%($\pm$0.45)
$0 \rightarrow 0.25 \rightarrow 0$ (gradually)	90.16%($\pm$0.37)	92.13%($\pm$0.33)	86.52%($\pm$0.37)
$0.25 \rightarrow 0 \rightarrow 0.25$ gradually)	87.62%($\pm$0.38)	90.43%($\pm$0.33)	80.96%($\pm$0.39)
$0 \rightarrow 0.25$ (suddenly at $t = 2000$)	90.00%($\pm$0.35)	91.85%($\pm$0.32)	87.34%($\pm$0.37)
$0.25 \rightarrow 0$ (suddenly at $t = 2000$)	87.13%($\pm$0.37)	90.72%($\pm$0.32)	80.44%($\pm$0.40)

TABLE A.2: Profit for dynamic closed organisations with similar tasks ($NoP = 5$)

SP variance	wolf-decay-Adapt (exponential)	wolf-decay-Adapt (linear)	wolf-k-Adapt
$0 \rightarrow 0.25$ (gradually)	72.98%($\pm$1.10)	74.78%($\pm$1.01)	69.99%($\pm$1.07)
$0.25 \rightarrow 0$ (gradually)	75.60%($\pm$1.04)	78.80%($\pm$1.01)	67.73%($\pm$1.04)
$0 \rightarrow 0.25 \rightarrow 0$ (gradually)	74.39%($\pm$1.07)	76.52%($\pm$1.04)	70.81%($\pm$1.05)
$0.25 \rightarrow 0 \rightarrow 0.25$ gradually)	72.69%($\pm$1.03)	76.27%($\pm$0.99)	66.36%($\pm$1.03)
$0 \rightarrow 0.25$ (suddenly at $t = 2000$)	74.82%($\pm$1.05)	76.32%($\pm$1.02)	71.38%($\pm$1.02)
$0.25 \rightarrow 0$ (suddenly at $t = 2000$)	78.88%($\pm$1.00)	82.11%($\pm$0.97)	71.15%($\pm$0.98)

A.4 Exponential Decay Methods

As mentioned in Section 5.1.1, the decay function used by decay-Adapt and wolf-decay-Adapt is linear. Here, we present experiments conducted on wolf-decay-Adapt when it uses an exponential decay function. In particular, Table A.1 and Table A.2 compare the performance of wolf-decay-Adapt when it uses linear decay and exponential decay function for dynamic closed organisations with dissimilar and similar tasks respectively. As argued in Section 5.1.1, the decay functions are dependent on other parameters like window-size and decay constant, and when these are set appropriately, there is no marked difference in their performance. Observing the results obtained, we find that the performance of wolf-decay-Adapt with an exponential decay function is close to that obtained by using a linear decay function. More importantly, it performs significantly better than wolf-k-Adapt for all the settings.

A.5 Summary

In this appendix, we have experimentally verified our assertions mentioned in Chapter 5. Specifically, Section A.1 showed that the initial structure of an organisation does not affect the performance of the adaptation process. Section A.2 helped us verify that the distributions used for the start-times and life-times of the temporary agents is not a significant parameter for evaluation. Following that, Section A.3 verified that the performance trends of the comparison methods remain the same even when the number of agents in the organisation is increased greatly. Finally,

Section A.4 showed that an exponential decay function performs almost as well as a linear decay function in the case of dynamic organisations.

Appendix B

Glossary

1. **Service:** We define a service as a specialised atomic action that can be executed by an agent. S is defined as the set of services provided by an organisation.

2. **Service instance (also referred to as 'SI'):** An instance of a specific service has two parameters. It specifies the type of service and the amount of computation required. Hence, we define a service instance to be $s_i = \langle s_i, p_i \rangle$ where $s_i \in S$ and $p_i \in \mathbb{N}$ denotes the amount of computation required. SI_w denotes the set of all service instances of task w.

3. **Dependency:** A service instance is dependent on another service instance if the execution of the former can only start after the completion of the latter.

4. **Dependency links:** The set of dependency links H_w contains links between the various si_j of the task w. These links are directed arcs between any two service instances depicting a sequential dependency from the source to the destination. An element h_j of H_w is of the form: $h_j = \langle si_a, si_b \rangle$ where si_a and si_b are the origin and destination of the link.

5. **Task:** A task is composed of a number of service instances with a precedence order. It is defined as a tuple containing a set of service instances and a set of dependency links between the service instances. $w = \langle \{si_i \in SI_w, \}, H_w \rangle$ (Equation 3.1).

6. **Agent:** Agents are independent computational entities capable of providing services. Every agent has two parameters, a set of services that it can provide and a computational capacity. Let A be the set of agents. Every element of A is of the form $a_x = \langle S_x, L_x \rangle$ where $S_x \in S$ denotes the services set of the agent and $L_x \in \mathbb{N}$ represents the computational capacity (Equation 3.3).

7. **Computational capacity:** This is defined as the maximum computational load that an agent can undertake in a single time-step. It is denoted by L_x for agent a_x.

8. **Stranger:** Two agents that are not aware of each other's presence are said to be strangers to each other.

9. **Acquaintance relation:** All agents whose presence is known to an agent constitute the acquaintances of that agent. Acquaintance relations are formally represented by the *Acqt* link in the organisation graph.

10. **Superior-subordinate relation:** An agent is the superior of another agent if the agent has a superior-subordinate relation with the other agent. The superior has a *Supr* link to its subordinate in the organisation graph.

11. **Peer relation:** Two agents are considered peers if they have a peer relation between them. Peer relations are formally represented by the *Peer* links in the organisation graph.

12. **Accumulated service set:** The accumulated service set of an agent is the union of its own service set and the accumulated service sets of its subordinates, recursively. For an agent a_x, it is denoted by $AccmSet_x$.

13. **Organisation graph:** The relationships between the agents in an organisation are represented in a organisation graph G. Every link g_i that belongs to G is of the form $g_i = \langle a_x, a_y, type_i \rangle$ where a_x and a_y are agents that the link originates and terminates respectively and $type_i$ denotes the type of link and it can take values $\{Acqt, Supr, Peer\}$ representing the three types of relations possible (Equation 3.5).

14. **Organisation:** The Organisation consists of the set of agents and a set of organisational links. Therefore, it can be represented by a 2-tuple of the form $ORG = \langle A, G \rangle$ where A is the set of agents and G is the set of directed links between the agents (Equation 3.4).

15. **Communication cost coefficient:** This denotes the cost in terms of the amount of resource used by the network to transmit one message between two agents. It is denoted by C

16. **Management load coefficient:** This denotes the amount of computation that is spent by an agent for evaluating whether a particular agent could be assigned a particular service instance. It is denoted by M

17. **Reorganisation cost coefficient:** This represents the cost incurred by the organisation while changing a relation between two agents. It is denoted by D

18. **Reorganisation load coefficient:** This represents the amount of computational units consumed by an agent while reasoning about adapting a single relation and is denoted by R.

19. **Cost of the organisation:** The cost of the organisation is the total amount of computational units utilised by the network for the transmission of the messages between the agents and the amount of reorganisation that has taken place (Equation 3.6).

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

20. **Load on an agent:** The load on an agent is the summation of the computational resources used by all the actions it performs in a time-step. The load l_x on agent a_x in a given time-step is (Equation 3.7):

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

where— (i) p_i is the amount of computation expended by a_x for executing SI si_i, (ii) $m_{j,x}$ is the number of relations considered by a_x while allocating SI si_j, (iii) W_{x_E} is the set of SIs (possibly belonging to many tasks) executed by a_x in that time-step, (iv) W_{x_F} is the set of SIs being allocated by a_x in that time-step and (v) r_x is the number of agents considered by a_x for adaptation, in that time-step.

21. **Reward from a task:** The reward obtained from completing a task w depends on the amount of computation needed by it and the speed of completion. It is calculated as (Equation 3.9):

$$reward_w = b_w - (t_w^{taken} - t_w^{reqd})$$

where t_w^{taken} represents the actual time taken for completing the task, while t_w^{reqd} is the minimum time needed and b_w is the sum of the computation amounts of all its SIs:

$$b_w = \sum_{i=0}^{|SI_w|} p_i$$

22. **Reward to the organisation:** It denotes the overall reward gained by the organisation for completing the tasks. It is calculated as the sum of the rewards obtained for the individual tasks (Equation 3.10):

$$reward_{ORG} = \sum_{w \in W} reward_w$$

where W is the set of tasks faced by the organisation.

23. **Profit of the organisation:** It denotes the overall performance of the organisation. It is denoted as (Equation 3.11):

$$profit_{ORG} = reward_{ORG} - cost_{ORG}$$

24. **form_subr**$_{x,y}$: It denotes the action of forming a superior-subordinate relation between agents a_x and a_y, with a_x as the superior.

25. **rem_subr**$_{x,y}$: It denotes the action of dissolving a superior-subordinate relation between agents a_x and a_y where a_x was the superior.

26. **form_peer**$_{x,y}$: It denotes the action of forming a peer relation between agents a_x and a_y.

27. **rem_peer**$_{x,y}$: It denotes the action of dissolving a peer relation between agents a_x and a_y.

28. **Assignment:** By assignment of a service instance to an agent, we mean that the agent has been allocated that service instance. It must accomplish that service instance by either executing it itself or by assigning it again to some other agent. Formally, when a service instance si_i is assigned to agent a_x, then a_x is considered an *assigned agent* for si_i.

29. **Delegation:** By delegation of a service instance to an agent, we refer to the fact that the agent is executing the particular service instance by itself (without reassigning it to someone else). Formally, when an agent a_x executes a service instance si_i, it is considered as the *delegated agent* for si_i. Thus, there could be several assigned agents for a particular service instance, but only one delegated agent.

30. **Value function:** It denotes the function that gives the values of the expected utilities of performing a reorganisation action between agents a_x and a_y. It depends on the estimated change in the load and costs of the two agents in question and the other agents involved in the delegations between these two agents. It is calculated as (Equation 4.1):

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

31. $\mathbf{Asg}_{x,y}$: The number of SIs assigned by an agent a_x to a_y. Assignment of a SI si_i by a_x to a_y means that a_x required that si_i be executed (was assigned to a_x or forms dependency of a SI executed by a_x) and it reallocated si_i to a_y. Thus a_y will have to be a subordinate, peer or a superior of a_x. Also a_y need not necessarily execute si_i itself, it could reassign it to one of its own subordinates, peers or superiors.

32. $\mathbf{Del}_{x,y}$: The number of SIs delegated by an agent a_x to a_y. Delegation of a SI si_i by a_x to a_y means that a_x is the agent that first required that si_i be executed (as it formed a dependency of a SI executed by a_x) and a_y is the agent that finally executed si_i (that is, a_y is the delegated agent). Note that, a_y may just have an acquaintance relationship with a_x. The delegation is always achieved through one or more assignments.

33. $\mathbf{t}_x^{tot}$: The total number of time-steps that a_x has been in existence.

34. $\mathbf{t}^{subr}x,y$: The number of time-steps that a_x and a_y had a superior-subordinate relation (that is, time-steps that $\langle a_x, a_y, Supr\rangle \in G$) . Likewise, $t_{x,y}^{peer}$ denotes the amount of time that a_x and a_y had a peer relation.

35. $\mathbf{filled}_x(t)$: The number of time-steps out of the total time denoted by t that a_x had waiting tasks (capacity being completely filled by load). The variable t can represent the total time of a_x (t_x^{tot}) or the time duration that a_y was its peer ($t_{x,y}^{peer}$) and so on.

36. $\mathbf{Asg}_{x,subr}$: The number of SIs that have been assigned by a_x to any of its subordinates. Likewise, $Asg_{x,peer}$ and $Asg_{x,supr}$.

37. $\mathbf{Asg}_{x,tot}$: The total number of SIs that have been assigned by a_x to other agents. Therefore, $Asg_{x,tot} = Asg_{x,subr} + Asg_{x,peer} + Asg_{x,supr}$.

38. $\mathbf{Asg}_{x,y}^{LOAD}$: The management load added onto a_y because of assignments from a_x (the count of these assignments is denoted by $Asg_{x,y}$ as stated above).

39. $\mathbf{IA}_{x,y}$: The total number of times, other agents (intermediate agents) were involved in the delegations of SIs by a_x to a_y. Therefore,

$$IA_{x,y} = \sum_{i}^{Del_{x,y}} count_{OA}^{i}$$

where $count_{OA}^{i}$ is the number of other agents involved in the delegation of si_i from a_x to a_y.

40. $\mathbf{IA}_{x,y}^{COST}$: The communication cost due to the delegations from a_x to a_y. For every agent in $IA_{x,y}$, a cost of $2C$ is added because a message is once sent forward and once back. Therefore, $IA_{x,y}^{COST} = IA_{x,y} * 2 * C$.

41. $\mathbf{IA}_{x,y}^{LOAD}$: The overall management load put on all the intermediate agents involved in the delegations from a_x to a_y (that is, $Del_{x,y}$). The load values are reported back to a_x along with the assignment information (the assignment message). If an intermediate agent has available capacity (no waiting tasks), it will report a 0 load value for that delegation. Otherwise, the agent will report the actual management load that was put on it due to that SI assignment.

42. **`k-Adapt:`** The fundamental method (without the subsequent enhancements) presented in Section 4.1.

43. **`wolf-k-Adapt:`** The extension to the fundamental method, based on the WoLF principle, as discussed in Section 4.2.1.

44. **`decay-Adapt:`** The modified method, using the decaying weights technique, presented in Section 4.2.2.

45. **`wolf-decay-Adapt:`** This method incorporates both the WoLF principle and the decay mechanism, that is Sections 4.2.1 and 4.2.2 respectively.

46. **`Central:`** It denotes the method in which the agents use an external, omniscient, instantaneously accessible and infinitely-resourceful centralised allocator to perform their task allocations.

47. **`Random:`** It denotes the method in which the agents adapt their relations in a random fashion.

48. **`free-Adapt:`** This refers to the method in which the reorganisation load coefficient R is always set to 0. It is the same as `k-Adapt` but all the acquaintances are chosen for reorganisation instead of a limited number. This makes it a theoretical upper bound for the performance of `k-Adapt` .

49. **`all-Adapt`:** Same as `free-Adapt` , differing only in R, which is set to the same value as in `k-Adapt` and not 0.

50. **Service Probability:** It denotes the probability with which a service is assigned to an agent. It is denoted by SP

51. **Patterns:** These are stereotypical task components used to represent frequently occurring combinations of service instances and dependency links. Similar to tasks, patterns are composed to service instances and dependency links, but are generally smaller in size.

52. **Number of Patterns:** It denotes the number of patterns used to compose the tasks in the task set W. It is denoted by NoP

Bibliography

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)', IFAAMAS, Honolulu, pp. 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 2007', Honolulu, USA, pp. 92–103.

Ashri, R., Luck, M., and d'Inverno, M. (2003), On identifying and managing relationships in multi-agent systems, *in* 'Proceedings of Eighteenth International Joint Conference on Artificial Intelligence', Acapulco, Mexico, pp. 743–748.

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.

Beverly, R. and Afergan, M. (2007), Machine learning for efficient neighbor selection in unstructured p2p network, *in* 'USENIX Tackling Computer Systems Problems with Machine Learning Techniques (SysML'07)', Cambridge, MA, USA.

Biskupski, B., Dowling, J. and Sacha, J. (2007), 'Properties and mechanisms of self-organizing manet and p2p systems', *ACM Transactions on Autonomous and Adaptive Systems* **2**(1), 1.

Bollen, J. and Heylighen, F. (1996), 'Algorithms for the self-organisation of distributed, multi-user networks. possible application to the future world wide web', *Cybernetics and Systems '96, Austrian Society of Cybernetics* pp. 911–916.

Bongaerts, L. (1998), Integration of Scheduling and Control in Holonic Manufacturing Systems, PhD thesis, PMA/K.U. Leuven.

Bou, E., Lopez-Sanchez, M. and Rodriguez-Aguilar, J. A. (2006*a*), 'Norm adaptation of autonomic electronic institutions with multiple goals', *International Transactions on Systems Science and Applications* **1**(3), 227–238.

Bou, E., Lopez-Sanchez, M. and Rodriguez-Aguilar, J. A. (2006*b*), Self-configuration in autonomic electronic institutions, *in* 'Coordination, Organization, Institutions and Norms in Agent Systems Workshop at ECAI '06', Trentino, Italy, pp. 1–9.

Bourjot, C., Chevrier, V. and Thomas, V. (2003), 'A new swarm mechanism based on social spiders colonies: From web weaving to region detection', *Web Intelligence and Agent Systems* **1**(1), 47–64.

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)', Seattle, USA, pp. 1021–1026.

Brazier, F. M. T., Kephart, J. O., Parunak, H. V. D. and Huhns, M. N. (2009), 'Agents and service-oriented computing for autonomic computing: A research agenda', *IEEE Internet Computing* **13**(3), 82–87.

Capera, D., George, J.-P., Gleizes, M.-P. and Glize, P. (2003), 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, Washington, DC, USA, p. 383.

Capera, D., Gleizes, M. P. and Glize, P. (2003), Self-organizing agents for mechanical design, *in* 'Engineering Self-Organising Systems', Vol. 2977 of *LNCS*, Springer, pp. 169–185.

Carley, K. M. and Gasser, L. (1999), Computational organization theory, *in* 'Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence', MIT Press, Cambridge, MA, USA, pp. 299–330.

Chapman, A., Micillo, R. A., Kota, R. and Jennings, N. R. (2009), Decentralised dynamic task allocation: A practical game-theoretic approach, *in* 'The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '09)', Budapest, Hungary, pp. 915–922.

Cohen, P. R. and Levesque, H. J. (1991), Confirmations and joint action, *in* 'Proceedings of the twelfth International Joint Conference on Artificial Intelligence (IJCAI'91)', Morgan Kaufmann Inc, Sydney, Australia, pp. 951–959.

Conitzer, V. (2008), Metareasoning as a formal computational problem, *in* 'Proceedings of the Workshop on Metareasoning: Thinking about Thinking at AAAI '08', Chicago, USA.

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', Banff, Canada, pp. 10–20.

Decker, K. and Lesser, V. R. (1993), 'Quantitative Modeling of Complex Environments.', *International Journal of Intelligent Systems in Accounting, Finance and Management. Special Issue on Mathematical and Computational Models and Characteristics of Agent Behaviour.* **2**, 215–234.

Deloach, S. A., Oyenan, W. H. and Matson, E. T. (2008), 'A capabilities-based model for adaptive organizations', *Autonomous Agents and Multi-Agent Systems* **16**(1), 13–56.

Di Marzo Serugendo, G., Gleizes, M.-P. and Karageorgos, A. (2005*a*), Self-organisation in multi-agent systems, Rapport de recherche IRIT/2005-18-R, IRIT, Universite Paul Sabatier, Toulouse.

Di Marzo Serugendo, G., Gleizes, M.-P. and Karageorgos, A. (2005*b*), 'Self-organization in multi-agent systems', *The Knowledge Engineering Review* **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, PhD thesis, Proefschrift Universiteit Utrecht.

Dignum, V. and Dignum, F. (2005), Structures for agent organizations, *in* 'International Conference on Integration of Knowledge Intensive Multi-Agent Systems', IEEE Computer Society, Waltham, USA.

Dignum, V., Dignum, F. and Sonenberg, L. (2004), Towards dynamic reorganization of agent societies., *in* 'Proceedings of the Workshop on Coordination in Emergent Agent Societies (WCES) at ECAI'04', Valencia, Spain, pp. 22–27.

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, Washington, DC, USA, pp. 128–135.

Ferber, J., Gutknecht, O. and Michel, F. (2003), From agents to organizations: An organizational view of multiagent systems, *in* 'Proceedings of the Fourth International Workshop on Agent Oriented Software Engineering (AOSE03)', Springer Verlag, Melbourne, Australia, pp. 214–230.

Fischer, K. (2005), Self-organisation in holonic multiagent systems, *in* 'Mechanizing Mathematical Reasoning', Vol. 2605, Springer, pp. 543–563.

Forestiero, A., Mastroianni, C. and Spezzano, G. (2008), 'So-grid: A self-organizing grid featuring bio-inspired algorithms', *ACM Transactions on Autonomous and Adaptive Systems* **3**(2), 1–37.

Fox, M. S. (1988), *An organizational view of distributed systems*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 140–150.

Galbraith, J. R. (1977), *Organization Design*, Addison-Wesley, Reading, MA.

Gasser, L., Braganza, C. and Herman, N. (1988), 'Implementing distributed AI systems using MACE', *Distributed Artificial Intelligence* pp. 445–450.

Gasser, L. and Ishida, T. (1991), A dynamic organizational architecture for adaptive problem solving, *in* 'Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI '91)', Anaheim, CA, pp. 185–190.

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, NY, USA, pp. 230–237.

Gershenson, C. (2007), Design and Control of Self-organizing Systems, PhD thesis, Vrije Universiteit Brussel.

Glinton, R., Sycara, K. P. and Scerri, P. (2008), Agent organized networks redux., *in* D. Fox and C. P. Gomes, eds, 'Proceedings of the 23rd AAAI Conference on Artificial Intelligence', AAAI Press, Chicago, pp. 83–88.

Grossi, D., Dignum, F., Dignum, V., Dastani, M. and Royakkers, L. (2006), Structural evaluation of agent organizations, *in* 'Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems (AAMAS '06)', ACM Press, New York, NY, USA, pp. 1110–1112.

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)', Vol. 1952 of *LNAI*, Springer-Verlag, Berlin, pp. 152–161.

Hassas, S., Di Marzo Serugendo, G., Karageorgos, A. and Castelfranchi, C. (2006), 'Self-organising mechanisms from social and business/economics approaches', *Informatica* **30**(1), 63–71.

Hilaire, V., Koukam, A. and Rodriguez, S. (2008), 'An adaptative agent architecture for holonic multi-agent systems', *ACM Transactions on Autonomous and Adaptive Systems* **3**(1), 1–24.

Hogg, L. M. and Jennings, N. R. (2001), 'Socially intelligent reasoning for autonomous agents', *IEEE Transactions on Systems, Man and Cybernetics, Part A* **31**(5), 381–393.

Holland, J. H. (1998), *Emergence: from chaos to order*, Addison-Wesley, Reading, MA, USA.

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, Hyderabad, India, pp. 1321–1326.

Hoogendoorn, M., Jonker, C. M. and Treur, J. (2007), Redesign of organizations as a basis for organizational change, *in* 'Coordination, Organizations, Institutions, and Norms in Agent Systems II (COIN'06 workshops)', Vol. 4386 of *LNAI*, Springer, pp. 46–62.

Horling, B., Benyo, B. and Lesser, V. (2001), Using self-diagnosis to adapt organizational structures, *in* 'Proceedings of the fifth international conference on Autonomous agents (AGENTS '01)', ACM Press, New York, NY, USA, pp. 529–536.

Horling, B. and Lesser, V. (2005), 'A survey of multi-agent organizational paradigms', *The Knowledge Engineering Review* **19**(4), 281–316.

Horling, B. and Lesser, V. (2008), 'Using quantitative models to search for appropriate organizational designs', *Autonomous Agents and Multi-Agent Systems* **16**(2), 95–149.

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, Berlin, pp. 506–515.

Ishida, T., Gasser, L. and Yokoo, M. (1992), 'Organization self-design of distributed production systems', *IEEE Transactions on Knowledge and Data Engineering* **4**(2), 123–134.

Itao, T., Nakamura, T., Matsuo, M., Suda, T. and Aoyama, T. (2002), Service emergence based on relationship among self-organizing entities, *in* 'Proceedings of the 2002 Symposium on Applications and the Internet (SAINT '02)', IEEE Computer Society, Washington, DC, USA, pp. 194–203.

Jackson, M. O. and Watts, A. (2002), 'The evolution of social and economic networks', *Journal of Economic Theory* **106**(2), 265 – 295.

Jelasity, M. and Babaoglu, O. (2005), T-man: Gossip-based overlay topology management, *in* 'Proceedings of Engineering Self-Organising Applications (ESOA'05)', Springer, Utrecht, The Netherlands.

Jin, Y. and Levitt, R. E. (1996), 'The virtual design team: A computational model of project organizations', *Computational & Mathematical Organization Theory* **2**, 171–196(26).

Kamboj, S. and Decker, K. S. (2006), Organizational self-design in semi-dynamic environments, *in* 'Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '06)', ACM Press, New York, NY, USA, pp. 335–337.

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)', Honolulu, USA, pp. 1220–1227.

Kephart, J. O. and Chess, D. M. (2003), 'The vision of autonomic computing', *IEEE Computer* **36**(1), 41–50.

Klein, F. and Tichy, M. (2006), Building reliable systems based on self-organizing multi-agent systems, *in* 'Proceedings of the 2006 international workshop on Software engineering for large-scale multi-agent systems (SELMAS '06)', ACM Press, New York, NY, USA, pp. 51–58.

Knabe, T., Schillo, M. and Fischer, K. (2003), Inter-organizational networks as patterns for self-organizing multiagent systems, *in* 'Proceedings of the 2nd international joint conference on Autonomous agents and multiagent systems (AAMAS'03)', ACM, New York, NY, USA, pp. 1036–1037.

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 (OAMAS) at AAMAS '08', Estoril, Portugal, pp. 1–16.

Kota, R., Gibbins, N. and Jennings, N. R. (2009*a*), 'Decentralised approaches for self-adaptation in agent organisations', *Submitted to ACM Transactions on Autonomous and Adaptive Systems* .

Kota, R., Gibbins, N. and Jennings, N. R. (2009*b*), A generic agent organisation framework for autonomic systems, *in* 'Proceedings of the 1st International Workshop on Agent-Based Social Simulation and Autonomic Systems (ABSS@Autonomics 2009)', Limassol, Cyprus.

Kota, R., Gibbins, N. and Jennings, N. R. (2009*c*), Self-organising agent organisations, *in* 'The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '09)', Budapest, Hungary, pp. 797–804.

Krackhardt, D. and Carley, K. M. (1998), 'A pcans model of structure in organizations', *Proceedings of the 1998 International Symposium on Command and Control Research and Technology* pp. 113–119.

Lematre, C. and Excelente, C. B. (1998), Multi-agent organization approach, *in* 'Proceedings of second Ibero-American Workshop on DAI and MAS', Toledo, Spain.

López Y López, F., Luck, M. and d'Inverno, M. (2006), 'A normative framework for agent-based systems', *Computational & Mathematical Organization Theory* **12**(2-3), 227–250.

Mainsah, E. (2002), 'Autonomic computing: the next era of computing', *Electronics & Communication Engineering Journal* **14**(1), 2–3.

Mamei, M., Vasirani, M. and Zambonelli, F. (2004), Self-organizing spatial shapes in mobile particles: The tota approach., *in* 'Engineering Self-Organising Systems, Methodologies and Applications (ESOA 04)', New York, USA, pp. 138–153.

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-Verlag, London, UK, pp. 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', Utrecht, Netherlands, pp. 25–29.

Mills, K. L. (2007), 'A brief survey of self-organization in wireless sensor networks', *Wireless Communications and Mobile Computing* **7**(7).

Miralles, J. C., López-Sánchez, M. and Esteva, M. (2009), Multi-agent system adaptation in a peer-to-peer scenario, *in* 'Proceedings of the 2009 ACM symposium on Applied Computing (SAC '09)', ACM, New York, NY, USA, pp. 735–739.

Mohamed, A. M. and Huhns, M. N. (2000), 'Benevolent agents in multiagent systems', *International Conference on Multi-Agent Systems* **0**, 0419.

Motwani, R. and Raghavan, P. (1995), *Randomized algorithms*, Cambridge University Press, New York, USA.

Nisan, N., Roughgarden, T., Tardos, E. and Vazirani, V. V. (2007), *Algorithmic Game Theory*, Cambridge University Press, New York, NY, USA.

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', *International Journal of Knowledge Based Systems* **17**(2-4), 103–111.

Picard, G. and Gleizes, M.-P. (2002), An agent architecture to design self-organizing collectives: Principles and application, *in* 'Adaptive Agents and Multi-Agents Systems', Vol. 2636, pp. 141–158.

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, Washington, USA, pp. 141–147.

Russell, S. J. and Norvig, P. (2003), *Artificial intelligence : a modern approach*, Prentice Hall series in artificial intelligence., 2nd edn, Prentice Hall, Upper Saddle River, N.J. ; [Great Britain].

Schillo, M., Bettina Fley and, M. F., Hillebrandt, F. and Hinck, D. (2002), Self-organization in multiagent systems: from agent interaction to agent organization, *in* 'Proceedings of the 3rd International Workshop on Modeling Artificial Societies and Hybrid Organizations (MASHO'02)', Aachen, Germany, pp. 47–56.

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, Honolulu, USA, pp. 1–8.

Sierra, C., Rodriguez-Aguilar, J. A., Noriega, P., Esteva, M. and Arcos, J. L. (2004), 'Engineering multi-agent systems as electronic institutions', *UPGRADE The European Journal for the Informatics Professional* **V**(4), 33–39.

Sims, M., Corkill, D. and Lesser, V. (2008), 'Automated organization design for multi-agent systems', *Autonomous Agents and Multi-Agent Systems* **16**(2), 151–185.

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, Melbourne, AUS, pp. 867–874.

Steels, L. (1990), Cooperation between distributed agents through self-organization, *in* 'Proceedings. IROS '90. IEEE International Workshop on Intelligent Robots and Systems '90. Towards a New Frontier of Applications.', IEEE Press, New York, USA, pp. 8–14.

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, Washington, DC, USA, pp. 464–471.

Thompson, J. D. (1967), *Organizations in Action: Social Science Bases in Administrative Theory*, McGraw-Hill, New York, USA.

Tumer, K. and Wolpert, D. (2004), A survey of collectives, *in* 'Collectives and the Design of Complex Systems', Springer, pp. 1–42.

Vazquez, L. E. M. and López Y López, F. (2007), An agent-based model for hierarchical organizations, *in* 'Coordination, Organizations, Institutions, and Norms in Agent Systems II (COIN'06 workshops)', Vol. 4386 of *LNAI*, Springer, pp. 194–211.

Vazquez-Salceda, J., Dignum, V. and Dignum, F. (2005), 'Organizing multiagent systems', *Autonomous Agents and Multi-Agent Systems* **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, Washington, DC, USA, pp. 129–132.

Watts, A. (2001), 'A dynamic model of network formation', *Games and Economic Behavior* **34**(2), 331 – 341.

Wolpert, D. H. and Tumer, K. (2001), 'Optimal payoff functions for members of collectives', *Advances in Complex Systems* **4**(2/3), 265–279.

Zambonelli, F., Jennings, N. R. and Wooldridge, M. (2003), 'Developing multiagent systems: The gaia methodology', *ACM Transactions on Software Engineering Methodologies* **12**(3), 317–370.

Zhong, C. and DeLoach, S. A. (2006), An investigation of reorganization algorithms, *in* 'Proceedings of the 2006 International Conference on Artificial Intelligence (ICAI 2006)', Vol. 2, Las Vegas, USA, pp. 514–517.