Content

List of Figures

List of Tables

List of Algorithms

Chapter 1: Introduction

Until recently, robots have been seen as a novelty. Today, the variety of robotic applications is growing at a tremendous rate and the trend will carry on in future as the progress in technology opens new possibilities in applications. A single-robot system is not an optimal solution for all applications. The growing range of existing and envisaged tasks that benefit from applications of multi-robot teams are, for example, search and rescue tasks, mapping of hazardous/hostile environments and space exploration/colonisation. However, the issue of coordination of multi-robot teams is not adequately resolved. To compound the problem, many robot architectures do not easily facilitate the implementation of coordination mechanisms. This thesis is aimed at contributing towards more efficient multi-robot teams, through development of a multi-robot architecture that facilitates coordination, as well as by proposing a new coordination mechanism.

1.1 Motivation

In the '80s and early '90s, robotic research focused on finding optimal robot architectures, often resulting in non-cognitive, insect-like entities. In recent years, processing power has improved and that, together with improvements in technology, has allowed for more complex robot architectures. Focus has thus shifted from single-robot to multi-robot teams. The key to the full utilisation of multi-robot teams lies in coordination. Unfortunately, many agent architectures are not designed with coordination in mind.

Although there are coordination mechanisms applicable to multi-robot teams, not one of them views a multi-robot team as a society. If a multi-robot team can be seen as a society, then some of the traditional society-based concepts (such as social networks) can be utilised for coordination.

Social networks are particularly attractive for application in multi-robot teams due to their emergent and self-organising nature. The new, social networks based approach

to coordination is envisaged for application to multi-robot teams; it is not robot-specific, and can be applied to any Multi-Agent System (MAS) without major modification.

1.2 The Objectives

There are two primary objectives of this thesis, both aimed at facilitation of coordination in multi-robot (and more general, MAS) systems:

- The development of a new agent architecture framework that facilitates implementation of coordination mechanisms. The emphasis is on robotic application and the architecture must utilise the best features of various robot architectures.

- The development of a new coordination mechanism that is applicable to multi-robot teams and MASs that operate in environments with a high degree of uncertainty.

Besides these two primary objectives, additional objectives of this thesis can be summarised as:

- The development of a simulated robotic environment, where experiments with various coordination mechanisms can be conducted.

- The full implementation of the proposed agent architecture framework in a physical environment, using a cheap, commercially available robotic platform.

1.3 The Main Contributions

The research effort that resulted in this thesis has achieved all objectives as stated in the previous section. The summary of the main contributions of this thesis can be stated as:

- A new flexible architecture framework for embedded agents was developed. The new framework, INDABA, can be seen as an extension of the currently predominant three-layer hybrid robot architectures with an additional layer that facilitates coordination. INDABA was successfully implemented in simulated and in real-world physical environments.

- A new coordination mechanism, through task allocation, was developed based on social networks. The agents in a MAS are treated as members of a society and social networks were used to determine agents' affinity to a particular type of task.

- A novel new way of implementing a complex agent architecture, such as INDABA, using a readily available, reasonably cheap, robotic platform. The novelty is that the architecture was easily split (due to its layered approach) into a component that resides in a PC and a component that resides in a physical robot. By doing this, the new architecture combined the processing power of a PC with a physical, real-world embedded robot.

1.4 Thesis Outline

This thesis is organised as follows. Chapter 2 provides a general background to agents, MASs and the origins of the agent paradigm. In addition, related issues such as interaction, coordination and cooperation between the agents in a MAS are also overviewed in chapter 2.

Chapter 3 focuses on robotics and three main agent architecture models, namely symbolic reasoning, reactive and hybrid agent architectures are overviewed. Each agent architecture model is firstly considered in a generalised manner, followed by a more detailed discussion of a particular, representative, agent architecture. The representative agent architectures are implemented in real-world robots.

The overview of agent architectures, given in chapter 3, is extended to multi-robot systems in chapter 4. The overview of multi-robot systems follows the format used in chapter 3. Two multi-robot architecture models are considered in generalised terms,

followed by a more detailed discussion of a particular implementation in a multi-robot team.

Chapter 5 introduces a new architecture, INDABA, that is designed for applications in multi-robot systems. Although designed with robotic applications in mind, INDABA is still general enough to be easily applied to any MAS. INDABA extends the currently predominant three-layer robot architectures by adding an additional layer that facilitates coordination.

Chapter 6 shifts focus from agent architectures towards coordination mechanisms that are used in MASs and multi-robot teams. The chapter starts with a brief overview of existing coordination mechanisms, followed by an introduction to the concept of social networks. Social networks are then applied as a coordination mechanism in a new coordination approach, which forms the main contribution of this thesis. The new social networks based approach is then applied to multi-robot teams.

The applicability of the new social networks approach is investigated in an abstract simulated environment in chapter 7. The agents in the abstract simulated environment were built around the INDABA framework. The results have confirmed the soundness of the social networks approach to coordination of abstract multi-robot teams.

The next step in confirming the social networks approach was to implement a more realistic multi-robot simulator environment. The results of experiments, together with the description of implementation of such multi-robot simulator environment are presented in chapter 8. Again, all simulated robots are built around the INDABA framework.

The final proof of soundness of any robotic architecture (or any of its components) is in its application in a real physical environment. This is achieved using the INDABA framework. The results of implementation of INDABA to a real robotic platform are presented in chapter 9. Furthermore, a social networks approach was applied to a scout selection process, and the results are described in the same chapter. Based on the results from application of the social networks approach to simulated environments (chapters 7 and 8), the assumption was made that the social networks

approach will perform well in a real, physical environment. The social networks approach is applied to a scout selection process in chapter 9. The results show that the social networks approach performs well in a real, physical environment.

Chapter 10 summarises this thesis and presents some directions for future research.

Chapter 2: Background

This chapter presents background on the agent paradigm and necessary definitions of what an agent and a Multi-Agent System (MAS) are. The chapter also presents an overview of the various origins of the agent paradigm. Current research is then described and a comparison is made between current approaches. An introduction and rationale for an agent system is given in section 2.1. Section 2.2 provides necessary definitions of an agent system as well as various classification methods for an agent. Section 2.3 extends agent systems into multi-agent systems and proposes a MAS classification method. Section 2.4 discusses problems related to MASs. Origins of agents and MASs, together with an overview of current research are given in section 2.5. Section 2.6 concludes this chapter with a summary.

2.1 Introduction

The research field of cooperating, embedded, heterogeneous multi-agent systems is becoming more mainstream than ever before. Many new MAS applications are simulated [173] and built [36]. This is hardly surprising, considering that the evolution of the paradigm for the development of computer systems has always lead to more independent, loosely-coupled modules. Initially, software development has relied on machine dependent, low-abstraction level, assembler programming. Procedural programming, as exemplified by 3rd generation programming languages (e.g. Pascal, C, etc.), was a major improvement on assembler-type programming languages. The onset of the object-oriented programming paradigm (e.g. C++, Java, etc.) heralded another qualitative shift towards more independent, reusable components. Today, mainstream information technology has fully embraced even more independent modules that interact through mechanisms such as CORBA, DCOM etc. Many researchers view agents as an extension of Component Based Software Engineering CBSE [151][81]. The evolution of CBSE can be illustrated as in figure 2.1.

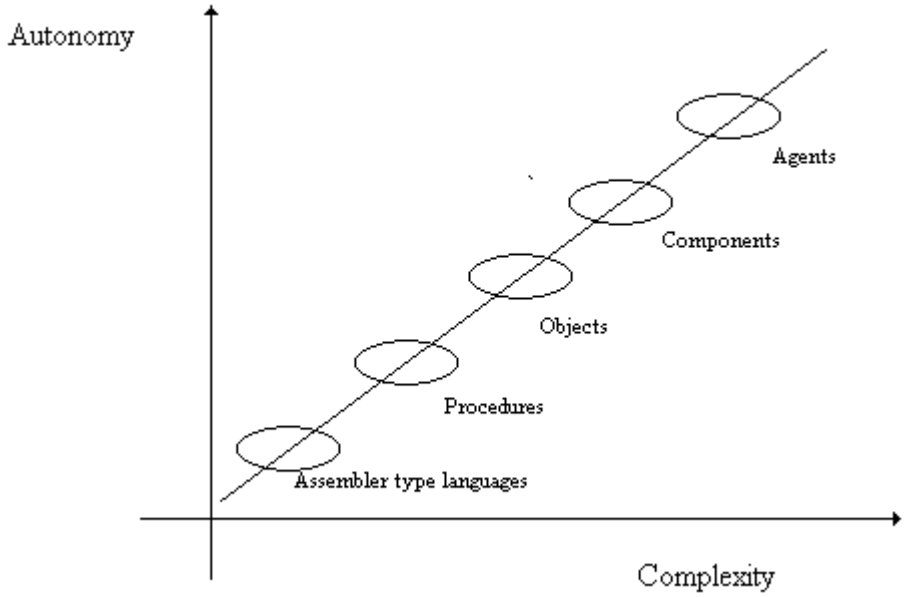

Figure 1. Evolution of Component Based Software Engineering

The current paradigm shift is towards independently interacting components that will have the property of self-organisation in order to solve a problem that is defined in general terms only. Those components are agents. Indeed, the probability is that most new IT development and products will, in one form or another, contain embedded agents [83]. The generalisation of agent systems resulted in the appearance of Multi-Agent Systems (MAS). MASs offer all the advantages of parallel distributed systems. The parallelism of MASs allowed application of the agent paradigm to an even greater set of problems that exceeded the capabilities of a single agent.

As the complexity of problems to be solved grew, so did the complexity of the paradigms that offer a solution to these complex problems. By introducing parallel distributed systems, the need for coordination between agents became obvious. The fact is that more and more complex Artificial Intelligence (AI) techniques are applied in the implementation of agents and MASs. AI techniques, as a general rule, are heuristics. It would be close to impossible to try to test a MAS using a white box approach. In other words, traditional tests and models are becoming rapidly obsolete as more AI based MASs are being developed and deployed. However, there is a need for some kind of model that will allow the computer scientist to design and test more complex systems. This has resulted in the recent, radically different approach to the

development of such models. As the systems are growing more complex, (nearly) approaching the complexity and diversity of simpler biological systems, a possible solution to the problem of moving towards the next paradigm is an AI-based MAS model using biological, social and organisational models. The idea is not exactly new [70][117], but recently it received momentum from the fact that some well-known researchers are proposing new MAS models based on social and behavioural models [141][53].

2.2 Agents: Definitions and Classifications

2.2.1 Introduction

Agent systems are rapidly becoming mainstream in the IT industry. The introduction to this chapter (see section 2.1) presented a reasoning for agent systems that mainly considered software engineering issues such as complexity hiding and the efficiency of parallel distributed systems. That is not the only reason for the increasing popularity of agent systems. The increase in complexity of tasks that are performed is not only imposed on the software systems developer. The complexity of tasks is affecting the end-users to an even greater degree, due to the fact that the user often has to perform a complicated set-up, and use complex operations in order to solve a problem. Considering the fact that computers are no longer viewed as tools for specialists only, the drive is to make efficient use of computers, even by inexperienced users.

One way of achieving this is to have intelligent helpers (agents) that help users to achieve a desired outcome. It is debatable if these intelligent helpers are agents in the true sense of agency (as proposed in section 2.2.3), as they have limits imposed on their autonomy, and collaboration between intelligent helpers and other agents systems (excluding users) is often limited. Nevertheless, proliferation of such agents is rapid. Some examples of such agent systems are Microsoft Office Assistant, Information Filtering Systems [200], intelligent web search engines [190][192] etc.

2.2.2 Agent Definitions

As is very common in the field of Artificial Intelligence, there is no standard definition of an agent. Instead, it seems that almost every major research and survey yields yet another definition. For the sake of completeness, some of the definitions are presented below.

An agent is:

"a computer system, situated in some environment, that is capable of flexible autonomous action in order to meet its design objectives" [197].

Others define an agent as:

- "a system that independently handles parts of the problem based on small independent knowledge bases" [82].

- "an autonomous entity that interacts with the environment, and adapts its state and behaviour based on interaction" [139].

- "an agent is a computational entity which:
  - acts on behalf of other entities in an autonomous fashion
  - performs its actions with some level of proactivity and/or reactiveness, and
  - exhibits some level of the key attributes of learning, cooperation and mobility" [80].

This thesis does not propose a new definition of an agent. Instead, an effort is made to extract the common characteristics of an agent from various agent definitions, as given in the next section.

2.2.3 Characteristics of Agents

As noted in the previous section, there is no common definition of an agent. However, it seems that most researchers agree on certain characteristics of agency. For the

purpose of this thesis, a computational entity is considered an agent if it possesses the following characteristics:

- **Autonomy**: An agent has its own beliefs, plans and intentions and it can accept or refuse a request.
- **Interaction**: An agent interacts with its environment. The agent can change the environment via its actions and the environment can change the agent's actions.
- **Collaboration**: An agent must be able to collaborate with other agents in order to achieve a common goal.
- **Learning**: An agent must have the ability to learn, based on previous experience from its interaction with the environment.

It is important to note that some of the quintessential agents and agent architectures do not fully have all of the proposed characteristics. Most notably, agents in the subsumption architecture [31] do not have full collaboration and learning characteristics, while agents in behaviour based architectures [113] do not "consciously" collaborate. The proposed set of characteristics can be seen as the result of evolution of the desired characteristics for an agent and represents the current mainstream approach to agency.

The prospects of having a standard definition of an agent are as good as having a standard definition of an intelligent system.

The next section presents some of the ideas that have contributed to the creation of an agent-oriented systems paradigm.

2.2.4 Agent Classification Schemes

Classification schemes for agents are relatively unexplored. Some classification schemes are implicitly given in various agent surveys [88][80] and some explicitly [138]. This thesis presents classifications based on agent reasoning model, agent key attributes [138] and agent paradigm origin. The following sections discuss each of these classification models.

2.2.4.1 Reasoning Model Classification

Classification based on an agent's reasoning method is not new. Despite the fact that classification based on reasoning method is not new, there is still no consensus on the exact naming of the two main paradigms that form the basis of this classification. The two main paradigms that form reasoning method classification are symbolic and sub-symbolic paradigms. Symbolic and sub-symbolic paradigms are respectively referred to as traditional and connectionist, or deliberative and reactive paradigms. These are all different names for the fundamental division between two different approaches in the field of AI.

According to reasoning method, agents can be classified into the following three distinctive groups:

- **Symbolic Reasoning Agents,** which utilise a traditional AI approach based on logic calculus. Traditional AI approaches are exemplified in the majority of expert systems. The main characteristic of a symbolic reasoning agent is that it relies on symbolic representation of the real-world. Symbolic reasoning agents usually have the following components [88]:

    - A symbolic model of the world, usually represented in some form of rules such as first-order predicate logic.
    - A symbolic specification of the agent's actions, usually represented as a rule with a condition for its triggering, which consists of an antecedent (a conjunction of Boolean conditions) and a consequent (or action).
    - A reasoning algorithm that plans the agent's future actions. All reasoning-related computations usually rely on inference rules, expressed in first-order predicate calculus.

    A detailed description and critique of symbolic reasoning agents is presented in chapter 3 (section 3.2), together with some examples of such systems.

- **Sub-symbolic Reasoning Agents,** which do not maintain a world model, or if they do, a non-symbolic representation is used for a world model. Sub-symbolic agents are sometimes called reactive agents. The main objective of sub-symbolic reasoning agents is to minimise the amount of predetermined behaviour, and to create agents that exhibit intelligent behaviour based on the agent's interaction with its environment. In other words, intelligent behaviour should emerge.

  The main characteristics of such agents are that they do not maintain a symbolic model of the world and usually do not communicate with other agents. The consequences are that a sub-symbolic agent's reasoning is based on interaction with the local environment.

  Despite the well-documented shortcomings of sub-symbolic agents [84][95], some of the sub-symbolic agent implementations have achieved spectacular results, albeit in very specific domains [30]. A more detailed description of this architecture and its critique is presented in section 3.3.

- **Hybrid Reasoning Agents,** which combine the characteristics of symbolic and sub-symbolic agents. Shortcomings of both symbolic and sub-symbolic models have become apparent fairly early and they are discussed in greater details in chapter 3. Various hybrid models were proposed that try to exploit the best of both approaches, such as MACTA [11][10], InteRRaP [130][129] and Touring Machines [63]. Most hybrid architectures are layered architectures, where lower layers are simpler (reactive or behavioural) and upper layers are more complex, providing symbolic reasoning capabilities, as well as mechanisms for cooperation between various agents.

This thesis assumes agent classification based on reasoning model.

2.2.4.2 Agent Key Attribute Classification

Nwana presents a typology based on the premise that agents can be classified along several ideal, primary attributes that the agent should exhibit [138]. The minimal set of identified attributes includes autonomy, learning and cooperation. If compared with

the desired characteristics of an agent, as presented in section 2.2.3, it is indicative that the characteristic of interaction with the environment is missing. The classification according to agent key attributes divides agents into seven distinctive groups:

- Collaborative (Cooperative) agents that are interested in cooperation with other agents. According to the agent's characteristics adopted in this thesis, all agents should be collaborative.
- Interface agents are agents developed to facilitate user-machine interaction.
- Mobile agents are agents capable of moving through physical environments, for example, robots.
- Information/Internet agents are agents mainly used for retrieval and search of information on the Internet.
- Reactive agents are agents that do not maintain any internal environment representation, and simply react on stimuli received from the environment.
- Hybrid agents that combine reactive agents with deliberative thinking.
- Smart agents were not clearly defined by Nwana but implicitly they should be "super-agents" combining collaboration, deliberative thinking and learning capabilities.

The shortcomings of the proposed classification are numerous but the classification is overviewed here for the purpose of completeness. The presented classification is a combination of divisions according to the agent's tasks and the agent's architecture and as such may lead to confusion as an agent can belong to more than one category (classified in one instance as what it does and in another instance as how it does it) according to this classification scheme.

Furthermore, some major categories are missing. For example, if the classification includes reactive and hybrid agents, it should surely include the symbolic reasoning (or deliberative) agents. Another such category is that of self-interested agents, as not all agents are collaborative. Due to the above-mentioned shortcomings agent key

attribute classification is of limited value for the purpose of this thesis, and it is not used in this thesis.

2.2.4.3 Paradigm Origin Classification

There were many contributing origins to the field of agent systems and MASs. Various overviews [88][138][80] have investigated the origins of agent paradigms. This section overviews a classification scheme based on a combination of these overviews.

Agents can be classified according to their original paradigm background into

- *Artificial Intelligence (AI) agents*, which is the main contributor to the field of agent systems [88]. Various sub-fields of AI have been incorporated into agent systems, such as artificial life, swarm intelligence, distributed artificial intelligence, traditional AI approaches and evolutionary computation. The AI contribution to current agent research is largely due to the scientific research done at academic institutions and various agents have their origins in AI research.

- *Object Oriented Programming (OOP) agents* – Many agent architectures are developed using the OOP paradigm [88][80]. It is not surprising that OOP is an origin of agent paradigm, considering that agents are the natural evolution of CBSE, as discussed in section 2.1. Frequently, objects are used as a starting point for an agent implementation because both agents and objects have shared characteristics, such as encapsulation and data hiding.

- *Machine-Man Interface agents* – Machine-man interface research is receiving strong impetus, based on industry and consumer demand. This is due to the continuous increase in complexity of tasks that today's and future users will face. Complex tasks need to be automated and streamlined. Agents are often used to help and guide users by being adaptive [80].

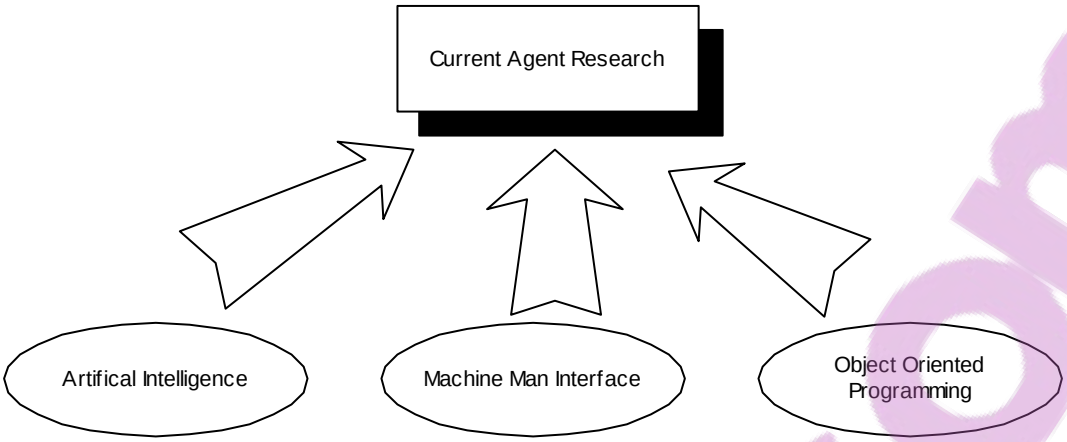

Figure 2. Origins of Agent Paradigm

- *Robotics* – Since the early age of civilisation, mankind was obsessed with creating tangible, real-world, intelligent, autonomous artefacts. Various explanations for such obsession can be given, but certainly some early inspirations, as described in literature, can be found in religion (e.g. the creation of golems, powered by the word of God [188]). Other reasons were economical (e.g. creation of the intelligent labourers as described in Karl Chapek's novel "Rossum's Universal Robots" [39]) and scientific (as in Mary Shelly's "Frankenstein" [169]). Today, robotics, the science of creating such artefacts, has come a long way from their literary and religious origins. Robotics, as a scientific discipline, often assumes a holistic approach to agent technology. It combines some of the disciplines above, such as software engineering, AI, artificial life, electronics, mechanics and other not so obviously related disciplines, such as organisational science, sociology, biology, etc. The product of robotics is a robot – the ultimate agent.

The origins of agent paradigm are illustrated in figure 2.

2.3 Multi Agent Systems: Definitions and Classification

2.3.1 Introduction

A system that consists of multiple agents is called a Multi-Agent System (MAS). A MAS is a generalisation of an agent system where the main advantages of agents can be further exploited, namely an agent's ability to execute both autonomously and in

parallel. A MAS is ideally suited for problems that can be either executed in parallel or that can employ multiple problem-solving methods. However, the advantage of a MAS approach to problem-solving and parallelism does come at a price: interaction problems between autonomous agents exists, including cooperation (working towards a common goal), negotiations (coming to an agreement) and coordination (avoiding harmful interactions between agents). Some definitions of MASs taken from literature are given in the next section.

2.3.2 MAS Definitions

There are various definitions of a MAS. For the purpose of this thesis only a few are presented. A MAS can be defined as a loosely-coupled network of problem-solvers that work together to solve problems that are beyond the individual capabilities or knowledge of each problem-solver [25].

Other authors keep the definition much simpler: a MAS can also be seen as a society of agents [204][80].

Wooldridge and Jennings propose a rather strict definition of a MAS that is based on MAS characteristics [88]:

- Each agent has incomplete information or capabilities for solving the problem, thus each agent has a limited viewpoint.
- There is no global system control.
- Data is decentralized.
- Computation is asynchronous.

2.3.3 Characteristics of MAS

This thesis proposes a slightly relaxed definition of MAS characteristics based on [88]:

- Each agent in a MAS can have complete or incomplete information about the problem or capabilities to solve the problem.
- There is no global rigid control system. However, there can be a global coordinating system, such as a supervisor.
- A complete set of data can be partially or fully decentralised.
- Computations are executed in parallel.

2.3.4 MAS Classification Schemes

Various classification schemes of MASs are in existence [88][138][80][160]. In this section, a subset of the existing MAS classification schemes is presented. Some of the presented classification schemes are generalised versions of agent classification schemes, while others are based on properties applicable only to MASs such as communication models.

2.3.4.1 Reasoning Model Classification

As is the case with agent architectures, MASs can be classified according to the reasoning module employed by the MAS. Using such a classification scheme, MASs can be divided into three classes: symbolic MAS, subsumption MAS and hybrid MAS. These classes are presented in chronological order of appearance.

Symbolic MAS

Symbolic architectures were the earliest to emerge as MAS [138]. This is hardly surprising if it is taken into consideration that a significant contribution to the agent paradigm came from AI planning research, which was a very active research area during the 1970s and 1980s. Symbolic MASs are based on premises of the "physical-symbol system hypothesis" [133]. Newell and Simon, defined a physical symbol system as a

"…physically realisable set of physical entities (symbols) that can be combined to form structures and which is capable of running processes that operate on those symbols according to symbolically coded sets of instructions" [133] .

The physical-symbol hypothesis then stipulates that a physical symbol system is capable of general intelligent action.

Wooldridge and Jennings define a symbolic architecture as

"[an] architecture that contains an explicitly represented, symbolic model of the world, and in which decisions are made via logical (or at least pseudo-logical) reasoning, based on pattern matching and symbolic manipulations" [197].

Symbolic architectures, as any other architecture, have their advantages and disadvantages, which are discussed in the section 3.2.2.

Typically, a symbolic MAS is based on a problem-solving method, such as STRIPS [65] that employs a symbolic, centralised model of the world. A symbolic MAS is based on the cognitive science sense-think-act cycle. The sense-think-act cycle assumes that an agent senses a change in the environment, deliberates about the change in the environment and decides on an optimal or nearly optimal course of action and, lastly, executes an action which may have an effect on its environment. In theory this sounds very good but there were problems when implemented in real-world environments. In practice, problems such as slowness of deliberation and accurate real-world modelling were experienced. Systems such as STRIPS [65] and General Problem Solver (GPS) [134] have performed extremely well in virtual worlds, where the model of the world was static and accurately given.

A more detailed discussion on the advantages and disadvantages of symbolic MAS is presented in section 4.3.

Sub-symbolic MAS

Once the limitations of symbolic MASs became obvious and theoretically proven [88], more criticism followed and the most influential critique came from Brooks [28][35]. As a complete opposite to the symbolic approach, an approach where knowledge is subsumed was proposed in the seminal work by Brooks [31]. In this purely reactive approach, knowledge is subsumed in condition-action pairs. Intelligence is treated as a "side-effect" of an agent's interaction with its environment.

The subsumption architecture employs no symbolic knowledge at all, hence there is no model of the world. It assumes that intelligent behaviour emerges from interaction between more primitive behaviours represented, essentially, as action-reaction pairs.

The subsumption architecture has been surprisingly successful, despite its apparent simplicity, but there are serious disadvantages of this architecture. An obvious problem is that, because of the lack of a world model, every agent decides on its actions based on information from its local environment. Therefore, there is no coordination as such, actions are only locally optimal, and overall behaviour is not easily understood.

An additional problem is that there is no effective way for an agent to learn from experience, as there is no direct feedback loop from consequences to actions. Details on this architecture are presented in section 4.4.

By the 1990s it was accepted that the subsumption architecture may be applicable to certain problem domains, such as modelling of insect behaviour [5], but it was not suited as a general architecture. An attempt to reconcile symbolic and sub-symbolic approaches resulted in the next class of MAS, i.e. the hybrid MAS.

Hybrid MAS

Hybrid MAS is a result of trying to use the best of both worlds, i.e. symbolic and subsumption MAS. Two main problems of the symbolic architecture, namely its slowness and the problem of accurate world modelling were related to its interaction

with its environment. Most of the strengths of the symbolic approach come from its deliberative and planned approach to acting on stimuli from the environment. On the other hand, the main strength of the sub-symbolic architecture stems from its efficient interaction with its environment and the main weakness is the fact there is no efficient, goal-driven interaction between agents. A typical hybrid system uses both symbolic and subsumed knowledge and exploits the strengths of each approach.

Typically, a hybrid MAS is a layered system, where different layers use different knowledge representations. The higher levels are based on symbolic knowledge reasoning, while lower levels are usually implemented using a sub-symbolic approach. A layered MAS exhibits symbolic planning and coordination, coupled with fast, efficient interaction with the environment. An example of a hybrid MAS is Multiple Automata for Complex Task Achievement (MACTA) [11][10] that utilises a symbolic planner as a symbolic component, while a sub-symbolic component is implemented using the Behavioural Synthesis Architecture (BSA) [106]. MACTA is presented in greater detail in section 4.5.

2.3.4.2 Cooperation Level Classification

With the appearance of MASs, the issue of avoiding negative interaction (or conflict) by means of negotiation and the issue of cooperation by means of coordination became very important. The potential for exhibiting negative interaction is due to the autonomy of agents, who may have their own beliefs, desires and intentions that are not necessarily shared between all agents. If agents' intentions are conflicting, a conflict may arise between the agents in a MAS. Classification of MASs based on the level of cooperation was proposed in the late '80s [24]. The cooperation based classification scheme has been adopted by other researchers [160][36]. According to level of cooperation between agents, MASs can be divided into:

Cooperative Multi-Agent Systems

Cooperative MASs, historically the first to appear, have their background in early Distributed Artificial Intelligence (DAI) [88]. In cooperative MASs, the emphasis is

not in optimising the performance of an individual agent, but that of the whole system. This class of MAS roughly corresponds to a symbolic MAS, as symbolic MASs often employ symbolic representation and cooperation enabling techniques that rely on symbolic representation of the world model. The consequence is that a global world model must be maintained.

The main focus of research in cooperative MASs is that of coordination between the agents [80].

Self-Interested Multi-Agent Systems

The emphasis of self-interested MASs is on improving performance of a single agent, hoping that improvement in the performance of an individual will lead to improvement in performance of the whole system. Unfortunately, agents may be openly antagonistic or they may exhibit conflicting behaviours. The problem is further compounded if there is no means of direct communication [113] or no communication at all [28].

When it comes to interaction between agents, the main areas of interest for self-interested MASs are that of conflict resolution and negotiation, assuming, of course, the existence of a communication channel between agents.

2.4 Problems with Multi-Agent Systems

The MAS paradigm is very promising, but it has its own problems that can be broadly divided into theoretical and practical problems. Theoretical problems relate to the interaction between the agents, while practical problems are related to the scalability to real-world environments and lack of formal methods and frameworks for agent development. Theoretical problems related to the interaction between agents are the main focus of this thesis and a new approach to coordination is presented in chapter 6 of this thesis.

Although the main focus of this thesis is on a specific coordination approach, the practical problem of inadequate frameworks for development of agents (and specially robots) is also addressed through the proposed new hybrid robot architecture (chapter 5).

2.4.1 Interaction Between Agents in MAS

Agents in a MAS perform their tasks in a shared environment. The agents not only interact with the environment, but with other agents as well. A simple example would be a robot scout that can detect an obstacle that can, for example, be either a wall (environment) or another robot (agent). The interaction between the agents in a MAS can be positive (resulting in cooperation) or negative (resulting in conflict). In order to address and facilitate cooperation, a MAS needs to have a coordinating mechanism. The problem of negative interaction between agents is very serious. To avoid and resolve conflicts, a MAS needs to have a negotiation mechanism. This section presents an overview of coordination and negotiation mechanisms.

2.4.1.1 Coordination Mechanisms

Cooperation allows agents in a MAS to solve problems that exceed an individual agent's characteristics. Coordination in a MAS is crucial to allow the exploitation of one of the main benefits of MASs, namely cooperation. Coordination is a problem not unique to computer science. The coordination problem is present in many different sciences, mainly in social sciences such as sociology, anthropology, organisational sciences etc. Coordination of biological systems such as ant colonies and swarms are studied in biological-related sciences [89], but also in certain AI fields such as swarm intelligence [61]. Coordination mechanisms can be classified according to their origin as follows:

Organisational Coordination

Probably the simplest way of coordinating agents is by establishing a relatively strict hierarchical architecture that prescribes roles and protocols for agents to communicate

with others within the hierarchy. Examples of organisational coordination architectures are MAGMA [181] and to a certain degree SMAK [93]. A shortcoming of the organisational coordination approach is that it may prohibit optimal task allocation due to extreme specialisation of each agent in the system. The hierarchical approach is also somewhat contrary to the idea of autonomous agents as they are coordinated from a central system and are not truly autonomous but dependent on the central system.

Contracting as Coordination

The contracting approach is based on an agent opening an auction for task allocation and other agents bidding for the executing the task. The idea of using an auctioning mechanism in AI is not new [79], with one of the most applied coordination techniques, the Contract Net Protocol (CNP), based on auctioning [49]. Although the authors, Smith and Davis, refer to CNP role as a negotiation tool, the view adopted in this thesis and by other researchers [88] is that it is really a coordination tool. The CNP assumes that agents fulfil separate roles, the one of bidder and the other of auctioneer, which has elements of an organisational coordination approach but the roles are not predefined and an agent can assume both roles. CNP offers a simple yet powerful, mechanism for coordination. The main critique of this approach is that it assumes a market economy [52], where there is an abundance of bidders and that the task should be relatively well known.

Society-Based Coordination

In view of one of the proposed definitions of a MAS as "a society of agents" [204] and in the recent work of some prominent researchers [141][53], it makes sense to let social regulations coordinate a MAS. Social regulations can be divided into social rules that regulate an individual agent's behaviour and social structures that regulate interaction between agents.

A more detailed discussion of coordination methods based on the theory of organisational sociology is presented in section 6.3.

2.4.1.2 Negotiation Mechanisms

The purpose of negotiation mechanisms is to prevent or resolve conflicts between the agents in MASs. There is still no consensus in the MAS research community on the importance of negotiation. While some architectures ignore negotiation completely [28], other researchers propose fairly complicated mechanisms for negotiation [151]. The proposed new approach presented in chapter 6 relies on coordination in order to prevent a problem instead on negotiation to resolve a problem. As the focus is on coordination, only an overview of negotiation techniques is presented in this thesis. Based on the classification given in 2.3.4.2, negotiation mechanisms can be divided into the following categories:

Competitive Negotiations

Competitive negotiations are particularly applicable to self-interested MASs where agents do not necessary cooperate; instead, the agents try to achieve their own goals. An example of a competitive negotiations environment is an agent trading in an e-commerce environment where negotiations (agreeing on a price) are done between self-interested, competing, autonomous agents [181][40]. There are various techniques used for competitive negotiations, such as game theory-related techniques [49], auctioning [181] and contracting [37].

Cooperative Negotiation

Cooperative negotiation mechanisms are applicable to cooperative MASs, where agents are willing to collaborate. This approach should be utilised when it is absolutely critical to avoid conflict, for example in the domain of air-traffic control [104][41]. The majority of systems that utilise cooperative negotiation are based on the Belief-Desire-Intention architecture [152], where negotiation is seen as updating and changing an agent's belief.

2.4.2 Scalability of MASs

Scalability of MASs to real-world problems can be viewed in many different ways. For example, it can be said that scaling up from a simulated stock exchange e-commerce trading system to a real-world stock exchange, a live e-commerce trading system can be a problem. After all, events in a real stock exchange environment are usually unpredictable.

Specifically, in the case of embedded agents (robots), scaling of systems that work very well in a simulated environment to real-world embedded agents has proven to be very difficult [34]. Initially, agents and MASs were built using a traditional symbolic approach to artificial intelligence. Although there were some success stories, such as STRIPS (Stanford Research Institute Problem Solver) which has been successfully tested on a real robot [65], it seems that symbolic reasoning cannot be the sole mechanism for the development of complex multi-agent systems [28][112]. The main reason for this is that deliberative, symbolic reasoning takes too much time in real-world environments due to the combinatorial explosion in potential decision-making processes in case of too many variables, which are usual characteristics of real-world environments.

2.4.3 Lack of Formalism

Because the agent oriented paradigm is still relatively new, research on standard design principles or standard frameworks is still limited [86][204][54]. While these approaches have been successful in standardising some of the explored domains, there is still no standard framework for the development of a MAS. The main reason why a unified framework does not exist is the variety of MASs and their application. Agents and MASs are applied in various domains ranging from Internet search agents (softbots), emulation of credible creatures in virtual reality [18] to envisaged interplanetary exploration [173]. A good example of the lack of formalism is the various means of communication mechanisms in MASs, ranging from simple 1Hz 6 byte broadcast messages [113] to the implementation of recommendations of the Knowledge Sharing Effort (KSE) group [132] that has resulted in sophisticated

models such as the Knowledge Query and Manipulation Language (KQML) [66][99] and Agent Communication Language (ACL) [166].

2.5 Origins of the Agent Paradigm

The concept of an agent did not appear suddenly. As indicated in the introduction, it can be said that agents evolved from the CBSE paradigm. However, CBSE was not the only origin of the agent paradigm. In fact, the agent paradigm has evolved from four main contributing fields, namely: Artificial Intelligence (AI), Object Oriented Programming (OOP), Machine-Man Interface Research and robotics as shortly discussed in section 2.2.4.3. Each of these origins is discussed next in more detail.

2.5.1 Artificial Intelligence

The aim of AI research is to produce intelligent artefacts. Intelligence is closely linked to the ability to learn. If these artefacts are situated in an environment and can interact with the environment, it seems that the natural aim of AI research is to produce agents. Nevertheless, agents have been ignored by mainstream AI for a surprisingly long period of time. A possible explanation for this anomaly is that AI researchers were too busy improving and investigating the AI components of a system (such as machine learning, or interaction mechanisms between software and its environment), without attempting a synthesis that would deliver a true agent system. An additional contributing factor is that AI, until relatively recently (as a point of reference, consider the re-emergence of Artificial Neural Networks (ANN) in the early eighties), was almost exclusively dominated by the symbolic reasoning paradigm as embodied in expert systems.

MASs are closely related, by virtue of being a collection of distributed, intelligent agents, to the field of Distributed Artificial Intelligence (DAI). DAI has been traditionally divided into two main groups [25]:

- Distributed Problem Solving (DPS), that considered how a problem can be solved by a number of modules that cooperate in dividing and sharing knowledge.
- Traditional MASs that are usually restricted to one type of replicated agents, also known as homogenous MASs.

A turning point was reached in 1980 at the first DAI workshop at MIT where it was decided that the aim of DAI is not to optimise low level parallelism issues, such as distributing workload between numerous processors or how to improve parallelism of algorithms, but to find how intelligent problem solvers can interact in order to solve problems that cannot be solved by a single intelligent problem solver [88].

2.5.2 Object-Oriented Programming

The similarities between an object and an agent are so obvious that it is not surprising that object-oriented programming (OOP) is one of the major origins of agent research. Both an agent and an object interact with its environment (via messages in traditional OOP), collaborate within the system (either using messages or methods) and, if considered learning in loose terms, both can learn (an object can "learn" by maintaining its internal state by means of protected and private data). These similarities are, however, misleading. There is a significant difference between agents and objects. Objects are not autonomous. Objects do not have their own goals, intentions and beliefs. In other words, the object represents an ideal body; the agent brings the reasoning. In a sense, the agent paradigm can be seen as a further evolution of OOP.

2.5.3 Man-Machine Interface

As the tasks that a user needs to perform on a computer become more complex, the way that a user interacts with a computer system becomes more time consuming and more cumbersome. Ideally, a user should just give the instructions on what he/she wants to achieve, without necessarily explaining in minute detail how to do this. For

this to be achieved new ways of interacting with computer systems are necessary. The man-machine interface research field is mainly interested in new ways of interacting with computer systems. One of the ways to streamline man-machine interfaces has lead to development of computer programs that cooperate with the user and that help the user to achieve what he/she really wants without explicitly instructing a computer system what to do. These computer programs need to be, at least partially, autonomous, need to interact, learn and collaborate. Such computer programs satisfy most of the characteristics of an agent. These agents are usually referred to as adaptive user interfaces or intelligent interface agents [80].

The tasks of intelligent interface agents can be divided into three groups based on the roles that the agents perform [80]:

Information filtering agents

The amount of, often unwanted, information presented to a user is increasing daily. This phenomenon is referred to as information overload. The role of an information filtering agent is to reduce the information overload based on user preferences. Filtering rules can be based on rules that the information filtering agents learn by "observing" the user's habits [190]. Alternatively, rules can explicitly be stated by the user, although the notion of agency in the second scenario is questionable. An example of an information filtering agent is Maxims [105] that manages user's emails and the user interface as implemented on amazon.com [190].

Information Retrieval Agents

The amount of information that is available for retrieval from the Internet is tremendous. It is no wonder that agents are employed in a role that allows a user to do an intelligent search of that vast amount of available information. Furthermore, agent controlled search can be executed in the background, collecting information from various sources and presenting results to the user, only when compiled and organised in a user-friendly format. An example of an information retrieval agent system is the Google search engine [192].

Expert Assistants

The task of expert assistants is to improve user interface efficiency by means of easier communication between the user and computer system. Expert assistants can be personified or not. Probably the most well known expert assistant is the Microsoft Office Assistant.

Adaptive user interfaces are not the only area of man-machine interface research that has contributed to the agent paradigm. Far more exotic than user interfaces and expert assistants are artificial life agents that populate virtual worlds in virtual reality man-machine interfaces, for example the Oz project at Carnegie Mellon University [65] and virtual worlds created in the MIT Media Lab [21].

2.5.4 Robotics

Wooldridge and Jennings do not consider robotics as an origin [197]. However, the description of a robot, as given by Chapek [39] (the author that coined the term "robot"), fulfils all four characteristics of an agent. Robots can be seen as agents and research in robotic architectures is a significant contributor to the agent paradigm.

The aim of robotics is to develop a machine that can assist humans. Robotics-related research of agent systems can be divided into two main groups, namely simulated robot systems and physical robots. These two groups are described next.

Simulated Robot Systems

Research in simulated robot systems is closely related to the field of AI and Distributed Artificial Intelligence (DAI). Distributed Problem Solving (DPS) is a sub-field of DAI, and in a way, multiple cooperating robots systems can be seen as a special case of distributed systems [38]. Simulated robot systems can be divided into two classes [113]: those that simulate situated agents and those that simulate abstract agents.

Simulated situated agents are embedded in simulated environments. One of the roles of simulated situated agents is that of a very valuable tool for making decisions on the design of physical robots. If the simulation environment accurately caters for physical laws and constraints, then design decisions can be made based on the results of simulations. Examples of design decisions are choice of sensors, sensor positioning, means of locomotion, etc. Other roles of simulated situated agents include accurate overall evaluation of a proposed physical robot system and experimentation on large-scale systems, which include a larger number of agents. A good example of a simulated robot system is given in [85].

Simulated robot systems that simulate abstract agents are useful for experimenting with aspects of robotic systems that are not related to robots' interaction with the environment, and as such have a limited role. Simulated robot systems that simulate abstract agents usually use a very high level of abstraction when interacting with the simulated environment. For example, a simulated robot system would assume that tasks such as "recognise-object" are atomic. From a DAI point of view, a simulated robot system with a high level of abstraction can be used to test cooperation and communications models, and in more general terms any biological and sociological aspect of MAS.

- *Physical Robot Systems*

Building physical robot systems as a MAS research vehicle is a substantial engineering task that, until recently, was attempted by a relatively small number of researchers. For the purpose of this thesis an overview, by no means exhaustive, of some of the seminal physical robot systems is presented.

Arguably, the earliest agent-related physical robot was Shakey, developed at Stanford Research Institute [153][136]. Shakey was equipped with the Stanford Research Institute Problem Solver (STRIPS) [65], a symbolic planner system. Various insights were gained; probably the most important is that mapping of a real-world environment to a symbolic world model is far from trivial. It has been observed that not all algorithms that perform well in simulated environments succeed in embodied systems. More details on Shakey can be found in [120].

In the ‘90s numerous robots based on the work of Rodney Brooks [31] were designed and implemented, for example, Myrmix [44]. Myrmix is a simple robot that has only three layers; each of the layers representing a simple action such as “collect”, “avoid obstacle” and “safe forward”. Genghis, also based on the work of Brooks [30], demonstrates how a simple architecture (the subsumption architecture) can achieve a relatively complex task, namely walking on six legs.

Behaviour based robotics, which developed from the subsumption architecture, addressed some of the shortcomings of the subsumption architecture. The shortcomings that were addressed include the lack of learning mechanism and lack of communicating mechanisms. Matarić, one of the foremost researchers in this field, has developed numerous physical robot systems based on the behaviour based robotics paradigm [113].

Another novel approach is that of the robotic ecosystem developed by McFarland and Steels [121], where the idea was to observe and facilitate emergence of cooperation between robots. Others, such as Aylett *et al* [11][10], created a hybrid architecture where a behaviour based architecture was implemented in robots and a symbolic planner component was implemented in a desktop computer. Arguably, the most important impetus to renewed interest in robotics research stemmed from the establishment of RoboCup [96]. Pfeiefer *et al* [148] have argued that the impetus that RoboCup has given to robotics can be compared to the impetus that the Apollo program gave to the exploration of space.

The renewed interest resulted in a large number of robot systems that have appeared over the last decade. Furthermore, the public interest in robotics has increased and as a result, numerous robotic kits are available today [185][184].

This thesis assumes learning and cooperation to be the key characteristics of an agent. One fairly recent physical robot system that emphasises these two characteristics deserves mention here, namely ALLIANCE [144] and its evolution, L-ALLIANCE [145]. Although its results do not exceed hand-crafted solutions, the system exhibited learning behaviour [142].

2.6 Summary

This chapter overviewed and discussed the various agent and MAS definitions as well as the origins of the agent and MAS paradigm. Some of the problems related to the MAS paradigm were also discussed. Various classification schemes were overviewed and the reasoning model classification scheme was adopted for the purpose of this thesis.

The next chapter provides a more detailed overview of agent architectures, classified according to the reasoning model used.

Chapter 3: Agent Architectures

Agent architectures can be classified according to various criteria (see section 2.2.4). For the purpose of this thesis, agent architectures are classified based on the reasoning model. In this chapter, an overview and a critique of each of the main classes of agent architectures are presented and discussed. A few definitions of agent architecture are given in section 3.1. Section 3.2 presents the historically first agent architecture: the symbolic reasoning agent architecture. Sections 3.3 and 3.4 discuss sub-symbolic agent architectures and hybrid agent architectures, respectively. Each of the presented architectures is initially discussed and criticised in its general form and then an example of such an architecture is described in greater detail. Section 3.5 proposes a hybrid agent architecture that is used in the INDABA agent architecture. Section 3.6 concludes this chapter with a comparison between the various presented models.

3.1 Introduction

The term agent architecture intuitively suggests a framework for the implementation of an agent. Agent architecture considers the issues surrounding the development and implementation of an agent based on a selected theoretical foundation. An agent architecture can be more formally defined as:

"[A] Particular methodology for building [agents]. It specifies how … the agent can be decomposed into construction of a set of component modules and how these modules should be made to interact…. An architecture encompasses techniques and algorithms that support this methodology" [106].

An alternative view on agent architecture is given as:

"[A] Specific collection of software (or hardware) modules, typically designated by boxes with arrows indicating the data and control flow among the modules. A more

abstract view of an architecture is as a general methodology for designing particular modular decomposition for particular tasks" [92].

There are various taxonomies proposed for agent and MAS architectures. The reader is referred to [38][90][57] for more details. In this thesis, agent architectures are classified according to the reasoning model used by agents.

In the remainder of this chapter, different agent architectures are presented and discussed.

3.2 Symbolic Reasoning Agent Architecture

Symbolic reasoning techniques are at the core of symbolic reasoning agent architectures. Section 3.2.1 presents a historical overview of the evolution of symbolic reasoning agent architectures, while section 3.3.2 presents some of the general characteristics and shortcomings of the symbolic reasoning approach.

As the representative of the symbolic reasoning agent architecture, one of the first implemented robots, namely Shakey [136], is discussed in section 3.2.3.

3.2.1 Introduction and History

Historically, the first agent architecture to appear, the symbolic reasoning agent architecture [136], has its roots in traditional artificial intelligence systems. An example implementation of a symbolic architecture is the early theorem-prover, General Problem Solver (GPS) [134]. Symbolic reasoning architectures are sometimes referred to as traditional architectures [87]. Expert systems are based on the symbolic reasoning paradigm. Successes of some of the symbolic reasoning systems, such as early expert systems (e.g. MYCIN [171]), have given credibility to the belief that such a paradigm can be easily extended to agent systems and embodied agents (robots). One of the seminal symbolic planning systems was STRIPS [65]. STRIPS is applied to agents and even to embodied agents, such as the robot Shakey [136]. Although the symbolic reasoning approach has been heavily criticised in the

late ‘80s (as discussed in section 3.2.2), the criticism did not stop the development of purely cognitive architectures such as SOAR [135] and ACT-R [4]. SOAR [135] and ACT-R [4] are both based on symbolic inference mechanisms. Agents systems that have utilised a symbolic planner as their main component include Integrated Planning, Execution and Monitoring (IPEM) [3] and “softbots” [62].

In the robotics field, after the initial success of Shakey [136] (which performed the required tasks, albeit slowly) and the harsh critique of symbolic reasoning systems during the ‘80s, there was not much development of pure symbolic reasoning, single agent systems. Similarly, the same applies to multi-robot symbolic systems. However, there were some simulated robotic systems based on a symbolic architecture, for example HOMER [193]. HOMER’s interaction with users was through commands that were given in a subset of the English language. Once commands were interpreted, the simulated robot would plan and execute given commands in its simulated environment.

It is becoming evident that any long-term artificial intelligence program must re-integrate some of the traditional AI based symbolic reasoning mechanisms [74]. The current trend is to create hybrid agent architectures where the symbolic component plays a significant role in agent architecture. An example of such a hybrid agent, 3T [22], is discussed in section 3.4.3.

The general characteristics of the symbolic reasoning agent architecture are presented in the next section.

3.2.2 General Characteristics of Symbolic Reasoning Agent Architectures

Symbolic reasoning agent architectures (also known as rational or deliberative) agents are based on a symbolic, abstract representation of the world and have an explicit knowledge base, often encompassing beliefs and goals [151]. Goal-oriented intelligent behaviour is explicitly coded into the agents, usually in production rule form. Typically, an agent can exploit many different plans to achieve its allotted goals. A plan is chosen on the basis of the current beliefs and goals of the agent. The

selected plan can be dynamically modified if these beliefs and/or the goals change. Rational agents can be considered advanced theorem-provers that manipulate symbols in order to prove some properties of the world. Implementing an agent as a theorem-prover allows the re-use of well-known techniques developed in the AI field, for example, the inference engines of expert systems.

The first obstacle that any symbolic reasoning architecture needs to overcome is that of an accurate implementation of the world model. The task of translating real-world entities and the often complex relationships between those entities into adequate symbolic representations is by no means a trivial task. Furthermore, there is no universal widely-accepted model for encoding the symbolic knowledge of the real-world. There are numerous methods in use, ranging from first order logic and production rules [87] to network representations, for example semantic nets [101].

The next problem that needs to be solved is which external stimuli, as sensed from the environment, can be ignored. Even for real embodied agents in real-world environments, information received via sensors is just a subset of all possible stimuli. For example, a robot may have a collision detector, but not a light sensor. The question that arises is whether a deliberative process will be different if a robot has more information about its environment. In other words, the choice of sensors that will influence the deliberative process is not always intuitive, and requires further research.

In real-world environments, symbolic reasoning suffers from the "real-time processing problem". To illustrate, consider that for real-world problems an optimal or nearly optimal solution is found based only on observation of the environment at initial time t. Most symbolic reasoning algorithms use a heuristic search of the problem space. However, because of a usually huge search space associated with real-world problems, an action executes during time t to $t+k$, where k is the time spent on finding an optimal or nearly optimal solution. In the meantime, during time k, the environment can change so that the optimal solution at time t may no longer be the optimal solution at time $t+k$.

Symbolic agent systems invariably have a lack of robustness to noise and inaccurate information [94]. In other words, symbolic agent systems do not degrade gracefully. This problem is common to most of the systems based on symbolic knowledge representation, for example, expert systems. As a result, symbolic systems usually perform well in simulated environments, but when implemented in a real-world environment, symbolic systems often fail to perform to their specifications.

One of the limitations of symbolic agents is that they execute sequentially. The sequential nature of a symbolic agent occurs due to the (essentially) sequential nature of the planning system that is at the core of symbolic reasoning architectures. Sequential execution of tasks may be acceptable for single agent systems, but in a MAS it is a serious shortcoming, due to the parallelism of MAS not being fully utilised.

Due to the shortcomings of pure symbolic reasoning approaches it is somewhat unlikely that pure symbolic reasoning architectures, on their own, will be predominant architectures of the future. However, symbolic reasoning techniques are widely used in the currently predominant architecture model, namely hybrid architectures (section 3.4).

3.2.3 Symbolic Reasoning Agent – Shakey the Robot

One of the first attempts at building a robot was a collection of hardware and software that was known as "Shakey the robot"[137]. Shakey used a symbolic planner, STRIPS [65], at its core. The next few sections discuss Shakey as an example of a purely symbolic reasoning agent.

3.2.3.1 Shakey – an Overview

Shakey was developed in the early '70s at Stanford Research Institute. The objective of the project was to incorporate vision, planning and the ability to learn into a single mobile robot. The main tasks that Shakey was designed to execute was to navigate from room to room and to push boxes, while avoiding obstacles.

Commands were given as "action routines" that operated at a very high level of abstraction. For example, the command Go_Thru (D1, R2, R1) meant "go from room R1 to room R2 via doorway D1".

Shakey used three sets of sensors:

- Bump detectors, implemented as touch sensors. The bump detectors were designed as antennas so that touch could be detected before the body of the robot touched an obstacle.
- An optical range finder, to provide Shakey with the distance from an object.
- A television camera together with image recognition software capable of recognising simple objects.

In addition to these sensors, Shakey also had a radio/video link to a stationary, off-board computer where the majority of processing was done.

Shakey was large by today's standards, having the size of an average-sized refrigerator and yet it had very little onboard intelligence [137]. Almost all processing was done on a mainframe using a radio link as a communication channel.

3.2.3.2 Shakey's Architecture

Intuitively, Shakey's architecture was well designed as it separated the actions performed into three groups based on "urgency" of the actions. Three action levels were implemented in Shakey's architecture, as illustrated in figure 3:

- Fast low-level actions (LLAs) that are represented by black lines on the figure 3. There are two types of LLAs. First type of LLAs are triggered by inputs from sensors without deliberation, similar to reflexes. Second type of LLAs are retrieval of rules and world model representation into a planner.
- Intermediate-level actions (ILAs), represented by broken grey arrows. ILAs represent simple, symbolic knowledge based actions.

- High-level actions (HLAs), represented by grey arrows. HLAs represent complex symbolic knowledge based reasoning, such as plans.

The soundness of Shakey's architecture was confirmed by the fact that Shakey could execute all of the envisaged tasks. Shakey could perceive its environment, plan and "reason" about its actions, and communicate. However, all of these tasks were executed excruciatingly slowly [35].

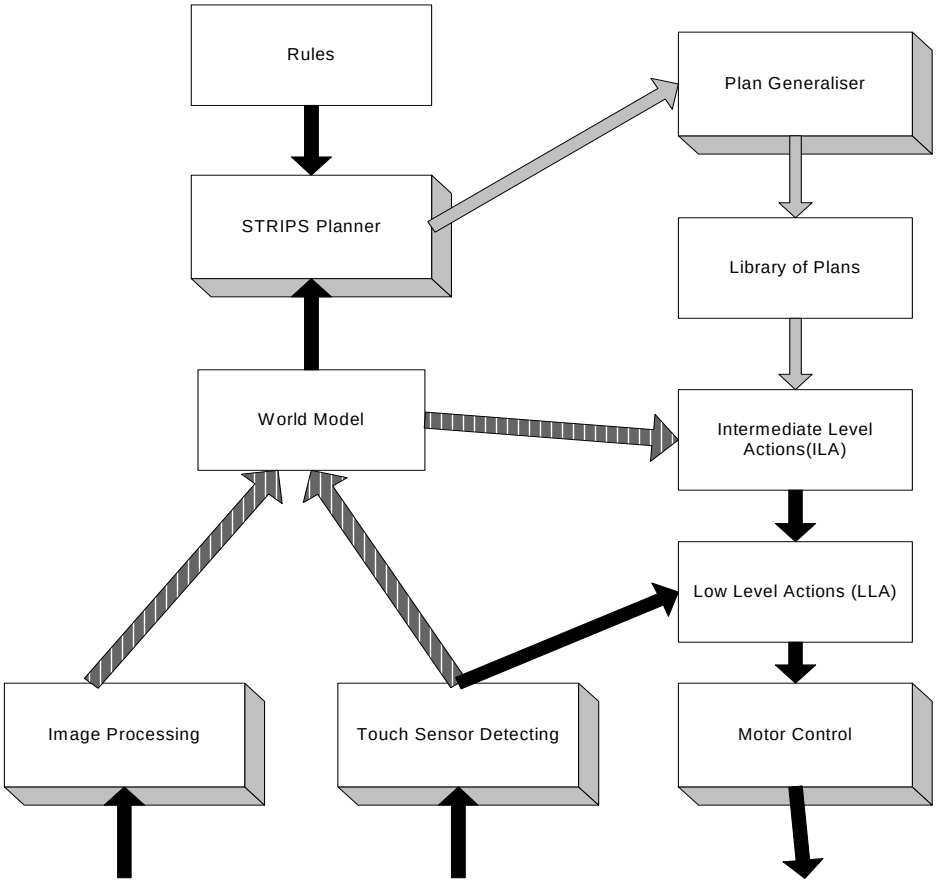

Figure 3. Shakey's Architecture, based on the description in [137]

The architecture is based on symbolic reasoning. All ILA and HLA actions are based on symbolic knowledge. However, it is interesting to note that the architecture included a sub-symbolic component as well. The creators of Shakey acknowledged a need for fast action-reaction couplings that are considered as reflexes. These "reflexes" are implemented as an LLA. An example of an LLA is that if touch sensors detect the proximity of an object, the touch sensor instantly sends a message to actuators (motors) to stop, without going through a symbolic reasoning deliberation process.

Although LLAs are executed without any deliberation, the symbolic representation of an actual situation is still created. The reason for creating a symbolic representation is to maintain a complete symbolic world model. For this purpose, the symbolic representation is sent to the upper levels that could deliberate upon the updated model.

Nilsson, one of the researchers that was involved in the Shakey project, refers to Shakey's architecture as a three-level architecture. Although the division of tasks is very similar to three-layer hybrid architectures that are described later in this chapter (section 3.4), Shakey's architecture should not be confused with hybrid three-layer architectures. In hybrid three-layer architectures there is a strong element of sub-symbolic based reasoning at lower layers, while Shakey's architecture is very much symbolic-oriented.

3.2.3.3 Shakey – Conclusion

Despite its limitations, such as slowness of its execution cycle, Shakey was a success. It performed the required tasks. More importantly, it gave significant insights on future development of robotic architectures.

The Shakey project has proven that:

"You could not, for example, take a graph-searching algorithm from a chess program and a hand-printed character-recognizing algorithm from a vision program and having attached them together, expect the robot to understand the world" [120].

Shakey has created a deeper understanding of problems associated with mobile, embodied agents. Shakey's approach to perceiving the world through its vision recognition software that could distinguish between boxes, walls and doorways was very advanced, especially considering the time period when Shakey was created.

One of the most important lessons learnt was that the creation of comprehensive world models was prohibitively computationally expensive. The majority of

computation was consumed in the transformation of sensor inputs to symbolic representations.

Other symbolic reasoning based robots were created, such as CART [127] and Hilare [78], but they all suffered from similar shortcomings, despite very simplified environments and the use of "state-of-the-art" symbolic reasoning mechanisms.

3.3 Reactive Agent Architecture

Not many papers have created such a reaction as the series of articles by Brooks [28][35][32] published in the early '90s. In these articles, Brooks delivered a harsh critique of the traditional AI approach to robotics. Brooks did not just criticise the traditional approach but also proposed, implemented and tested an alternative approach that has since become known as the subsumption architecture.

The subsumption architecture and its derivatives are often referred to as reactive architectures [130], a terminology adopted for the purpose of this thesis. Section 3.3.1 presents a historical overview of the evolution of reactive agent architectures, while section 3.3.2 presents some general characteristics of this approach. As the representative of reactive agent architectures, the original subsumption architecture [31] is discussed in section 3.3.3. A discussion of reactive agent architectures is given in section 3.3.4.

3.3.1 Introduction and History

Although a radical departure from then mainstream AI techniques, the subsumption architecture can be traced to another experiment from the mid-'80s. Braitenberg [26], as an experiment in cognitive science, proposed the development of 14 simple vehicles of varying characteristics. The first six vehicles had a very simplistic coupling of sensors to actuators and in a sense were very similar to the simple layers of the subsumption architecture. The subsumption architecture was proposed in 1986 [31] and it has been the inspiration for many other attempts and implementations [33][44][159][113]. Some implementations improved on the subsumption architecture

[44][159], while others used the subsumption architecture as the foundation that has led either to behaviour based robotics [113] or to hybrid systems such as [6].

Despite the fact that well-founded criticism had been levied against reactive agent architectures relatively soon after their appearances [84][95], further research in reactive architectures did not stop. Example applications of reactive architectures are that of Altenburg [2] (which is of special interest to this thesis as it is based on the same robotic platform as used in chapter 9), and Cog [33], again a project by Brooks and his team.

Although Cog initially exhibited some sophisticated behaviour, achieved through a basic reactive architecture, the Cog architecture had to incorporate some learning mechanism (implemented using neural networks) in order to achieve coherent behaviour [148].

Today, pure reactive architectures are not used in isolation due to the shortcomings that are presented in section 3.3.4. Reactive architectures have been superseded by behaviour based architectures, such as [113][68], not only in single agent systems, but also in hybrid systems [22][129].

3.3.2 General Characteristics

3.3.2.1 Origins of Reactive Architectures

Notwithstanding the similarity between Braitenberg vehicles [26] and their cognitive science approach, the major contributing origins of purely reactive architectures can be traced to two distinctive sources: biological sciences and engineering sciences.

From the biological sciences, the inspiration was drawn from the fact that the traditional notion of intelligence (i.e. the cognitive notion of intelligence) in biological systems has appeared very recently in evolutionary terms. The emergence of intelligent and cognitive thinking was preceded by millions of years of improving on

the interaction of biological systems with their environment. This gradual approach, which eventually results in emergent intelligent behaviour through interaction with environment and agents, was an inspiration to Brooks. Brooks focussed his research on the development of environment interaction mechanisms. Brooks states that:

"…mobility, acute vision and the ability to carry out survival-related tasks in a dynamic environment provide a necessary basis for the development of true intelligence" [28].

This view has been shared by other researchers [126][112]. From a biological perspective, the objective behind reactive architectures is then to create complete creatures that can exist in a dynamic people-populated world [34].

As such, creatures are dependent on efficient interaction with their environment. On the other hand, from an engineering point of view, it is imperative to develop efficient coupling between sensors and actuators, and the methodology that facilitates the development of such couplings.

The approach adopted for such couplings was uncompromisingly designed for speed and robustness. Unfortunately, the adopted approach was not sufficiently flexible and it has become the main reason for the limitations of reactive architectures, as discussed in section 3.3.3.3.

3.3.2.2 Underlying Concepts

Reactive architectures are based on the following fundamental underlying concepts:

- Situatedness: An agent is situated in its environment and directly interacts with the environment, without building a world model. This elegantly solves the problem of accurate world modelling and symbol grounding (as discussed in section 3.2.2). The agent is using the world as its model. In extreme interpretations of this architecture, the world is used even as a communication channel between the different layers of a reactive agent.

- Embodiment: Brooks argues that the only way to make sure that an agent can function in the real-world is if it has a physical body [28][35]. This view is shared for the purpose of this thesis.

- Intelligence: Reactive architectures adopt a bottom-up approach for intelligence modelling. Bottom-up approach means that basic layers are created first and then combined into more complex layers. Reasoning is also deemed unnecessary as it relies on a symbolic world model.

- Emergence: The main objective of a reactive agent architecture is that, through the agent's interaction with its environment, intelligent behaviour will emerge. It is very important to note that relatively simple agents that are not "aware" of their intelligence (they do not maintain any reasoning mechanisms) can exhibit emergent intelligent behaviour.

Brooks derives four ideas from each of the above concepts as an inspiration for the subsumption architecture [35]:

- "The world is its best model" – inspired by situatedness;
- "The world grounds regress" – inspired by embodiment;
- "Intelligence is determined by the dynamics of interaction with the world" – inspired by intelligence; and
- "Intelligence is in the eye of the observer".

The subsumption architecture is implemented as a set of layers that define agent behaviour. These layers are described next.

3.3.2.3 Layering in Reactive Architectures

The reactive architecture proposes building of simple layers based on augmented finite state machines. These layers are implemented as couplings between sensors and actions. All the layers execute in parallel and all the layers interact with the

environment. In other words, the reactive architecture is a horizontally-layered architecture.

The layering concept is demonstrated in figure 4. Three complex layers (the "collect" layer, "avoid obstacle" and "safe forward" layers) are implemented in the example. Only the simplest layer (the "safe forward" layer) is presented in detail. Layers that collect objects and avoid obstacles are abstracted.

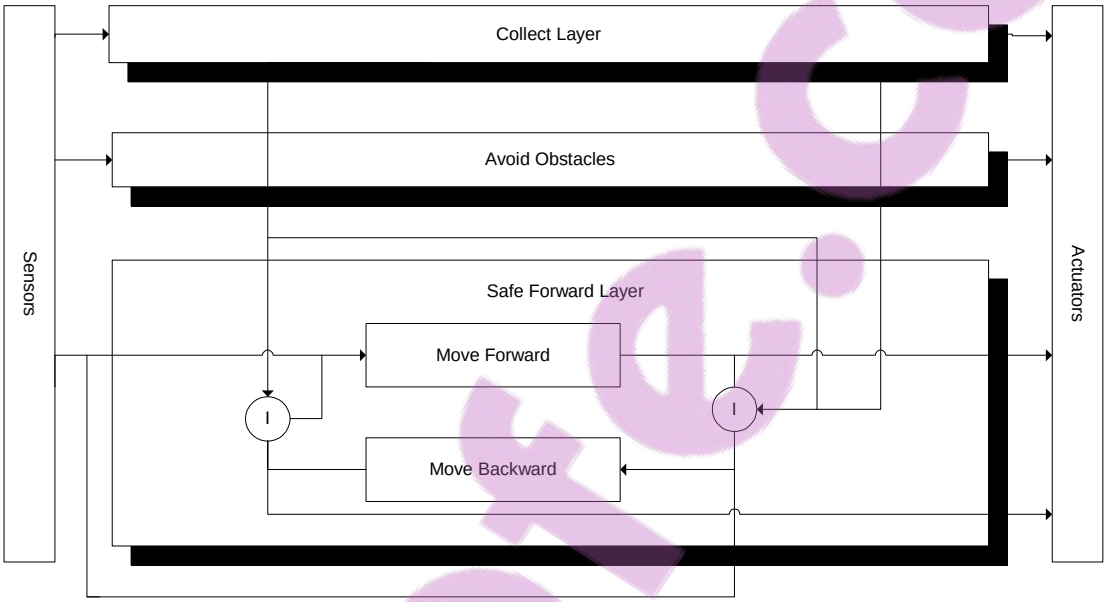

Figure 4. An example of the layers of a reactive agent

To illustrate the layering in reactive architectures, consider the simple actions implemented as layers "move forward" and "move backward". These two simple layers are directly coupled to actuators and sensors (see figure 4). In the reactive architecture, more complex actions are achieved through manipulation (inhibiting and enabling) of inputs and outputs of the appropriate layers. Examples of such layers are "avoid obstacle" and "collect" layers. The "avoid obstacle" and "collect" layers can enable and disable simpler layers such as "move forward" and "move backward".

Brooks advocates that the layering of a task-specific control can be achieved through this mechanism but it is unclear how this can be achieved without the decision on behaviours made at the time of design.

The inhibit and enable control mechanism is far from modular. More complex layers are tightly coupled with simpler layers. The consequence is that even a minor change at a simpler layer can have a severe consequence for the behaviour of a more complex layer and of the robot as a whole. Information-flow from the simpler layer to the more complex layer is non-existent.

The "safe forward" action is illustrated as follows: if the sensor input from the proximity detector is below the threshold (which is an indication that there are no obstacles), *move forward* is maintained and *move backward* inhibited. The moment the obstacle is detected, the situation gets reversed; *move forward* is inhibited and *move backward* initiated. The upper layer can influence lower layers as indicated in the figure.

3.3.2.4 Is a Reactive Agent Truly an Agent?

According to the definitions given in section 2.2.2, a straight answer cannot be given to this question. A reactive agent conforms to definitions as given in [197] [139], but not according to the definitions given in [82][80]. More importantly, considering the characteristics of agency as given in section 2.2.3, the answer is no. The first three characteristics of agency (autonomy, interaction and collaboration) can be (arguably) satisfied by a purely reactive agent, but the last characteristic, learning, cannot be satisfied.

In a purely reactive architecture there is no provision for any world model or internal state and therefore reactive architectures lack the basic fundamentals necessary for learning. Pure reactive agents do not learn.

3.3.3 Reactive Agent – Subsumption Architecture

Although it is arguably not a true agent architecture, the subsumption architecture is of such seminal importance, not only to the field of agent systems, but to the whole AI field, that it is discussed as an example of reactive architectures. Many robotic

systems were built based on the subsumption architecture and most of them were very successful [33][27][43].

Layers are implemented as finite state machines with four possible states:

- Output state. If a layer is in the output state, the layer outputs a message according to a computational transition and then switches to a predetermined state.
- Conditional Dispatch state. When a layer is in the conditional dispatch state, the layer tests the value of a function and then switches to one of the predetermined states.
- Self state. When in a self state, the layer performs a computation that affects the internal state (albeit limited to a few variable registers).
- Event Dispatch state. In the event dispatch state, a layer waits for event(s). Once an event occurs, the layer switches to a predetermined state.

Each layer has input and output that can be affected by suppressor and inhibitor connections respectively. If the inhibitor is active (a message going through the wire) then the outputs of the corresponding layer are suppressed. If the suppressor is active, then the input is disregarded. A typical black box representation of a layer is given in the figure 5.

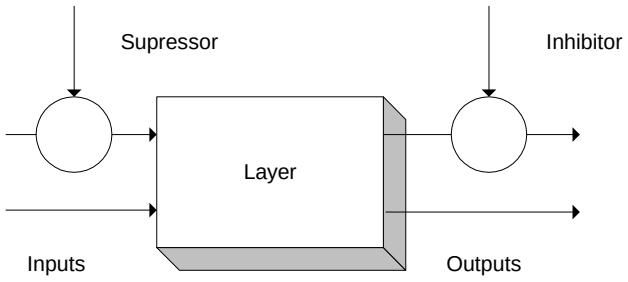

Figure 5. Black Box Approach for a Reactive Agent Layer

Suppressing inputs and inhibiting outputs of simpler behaviour achieves more complex layers. The whole methodology for building agents based on the subsumption architecture is based on the iterative approach as described next.

Simple layers are built and tested in the real-world environment. Once successfully tested, the simple layers serve as building blocks for more complex layers that are then tested in the real-world environment. Again, these layers now serve as building blocks for even more complex layers and the whole process is repeated until the desired behaviour is achieved.

A reactive architecture that are implemented according to the above described methodology conforms to the goals that are stipulated by [28]:

- The capabilities of the agent are built up in small incremental steps and at every step there is a complete system that can be tested.
- At each step, the embodied agent is tested in the real-world environment.

As indicated at the beginning of this section, there were some notable successes of this architecture. The reactive architecture has probably reached its pinnacle in the implementation of the robot, Herbert [29]. Herbert incorporated mobility, image recognition and robotic arm coordination. Its tasks were to wander around offices and to collect soda cans. According to Gat [74], Herbert has never reliably performed the desired task.

3.3.4 Subsumption Architecture - Conclusion

The subsumption architecture deviated from the traditional AI (symbolic reasoning) systems. The traditional AI systems usually implemented the symbolic reasoning mechanism, the Sense – Think - Act cycle [148].

There were many problems with the symbolic reasoning approach as discussed in section 3.2.2. The core of the problem lies in the drastic simplification of the symbolic model of the world, which is necessary to reduce the search space of the planner. Reactive architectures have eliminated this problem. Reactive architectures do not maintain a world model; instead, reactive architectures indirectly use the world as its own model.

Although the reactive architecture approach is good for simple agents and behaviours, it is, unfortunately, inadequate for more complex tasks mainly due to the lack of any world model. The lack of world model in the subsumption architecture has introduced the following problems:

- Agent reasoning is based purely on sensor readings from the local environment. This can lead to a local optimum solution (a course of action) that is not the global optimum solution (the best possible action).
- It is difficult to see any possibility for learning from experience or from other agents.
- The idea of emergence is valid, but reactive architectures do not provide mechanisms for recognition and incorporation of such emerging behaviour. Instead, all layers are handcrafted, so any new layer (behaviour) would require reprogramming.
- Interdependencies between the layers can become unmanageable in case of many layers, due to the numerous couplings between layers.
- With MASs in mind, the lack of direct communication between the agents (even between the lower layer to the upper layer) can lead to negative interaction, i.e. conflict. The lack of communication (except through the world itself) prevents implementation of any coordination mechanism.

Although the shortcomings of the subsumption architecture are numerous, the subsumption architecture, and reactive architectures in general, played a major influential role in today's embedded agent predominant architectures, namely the hybrid architectures. Due to the reactive architectures' superb interaction capabilities with the real-world, reactive architectures (or their derivatives) are often used as the simplest layer of hybrid architectures.

Reactive architectures should also be considered in relation to the time period when they appeared. The computational cost in terms of hardware has dropped tremendously in the last 20 years and computational power has increased almost exponentially. This development can influence the validity of the "do not think (because it is costly and time consuming) – act!" premise of reactive architectures.

Many robots [33][27][43][29] have been built based on the subsumption architecture, some of them very successful. In a sense, these early successes have created a renewed interest in robotics.

The root of the shortcomings of the subsumption architecture can be found in the definition of the subsumption architecture, as given by Brooks. Brooks defined the subsumption architecture as a parallel and distributed computation formalism for connecting sensors to actuators in robots [34]. The strong engineering influence in the development of the subsumption architecture is evident from this definition. In other words, the subsumption architecture is mainly concerned about hardware efficiencies without much concern for higher concepts such as models, learning capabilities and sociality between agents.

It can be claimed, with confidence, that without the subsumption architecture and the pioneering work done by Brooks and his team, robotics would be far from the capabilities that are demonstrated today.

3.4 Hybrid Agents

As discussed in sections 3.2 and 3.3, purely reactive and symbolic agent architectures have different shortcomings and benefits. It was natural that further research led to the emergence of hybrid agents which attempt to exploit the strong points of each approach.

Purely reactive architectures have been criticised for their inability to perform complex tasks [88]. Reactive architecture-inspired approaches, such as behaviour based robotics [81][109], have replaced purely reactive architectures. Section 3.4.1 tracks some of the hybrid agent history, and section 3.4.2 overviews the typical three-layer architecture that has become almost standard for hybrid agent architectures. Section 3.4.3 describes an example of hybrid agent architecture in greater detail.

3.4.1 Introduction and History

Once the critique of purely non-symbolic architectures appeared [84][95], various attempts were made on improving on non-symbolic architectures [159][165]. By the early nineties at least three teams of researchers [73][45][23] had independently proposed true hybrid architectures, consisting of three layers.

These early hybrid agent architectures have evolved over time, being continually improved. For example, Bonasso's work has evolved into the three-tier architecture, 3T [22] and Gat's has evolved into an architecture called ATLANTIS [73]. Both of these architectures have been successfully applied in robotic systems. 3T has been used as the core architecture for NASA Johnson Space Center's Robotic Architecture robot that was able to recognise people [22]. ATLANTIS was also implemented in a number of robotic systems [73][71].

InterRAP [129], another important hybrid agent architecture, combines not only reactive and symbolic planning aspects of agents but also the social aspects. It is also interesting to note that ATLANTIS and 3T have their origin in robotics, while InterRAP has its origins in DAI [90].

The more recently proposed hybrid architectures is Jet Propulsion Laboratory's (JPL) CLARAty [194]. The importance of CLARAty is that it proposes the use of existing methodologies, software and approaches such as open source software libraries and object-oriented design methodologies (e.g. the Unified Modelling Language (UML)).

In addition to the aforementioned architectures, there are also various design tools that support hybrid architectures. Two of the design tools that support hybrid architectures are mentioned here for the sake of completeness. DESIRE [58] is a methodology that provides basic modularisation techniques that can be used for building hybrid agent systems. DESIRE provides for horizontal and vertical layering approaches (as described in the next section). Task Description Language (TDL) is a software development tool [173] that is implemented in the C programming language as an extension, and can be used for the development of three-layer architectures.

3.4.2 General Characteristics of Hybrid Agent Architectures

In this section, some of the general characteristics of hybrid agent architectures are overviewed and briefly discussed.

3.4.2.1 Layered Architectures

Most hybrid systems are implemented using layered architectures. Architectures can be layered horizontally or vertically. In the horizontal layering approach all layers are at the same level and execute independently. Vertical layering means that there is a number of layers between the sensors and actuators, while in horizontal layering there is only one layer that has as inputs sensor inputs and as outputs actions. Brook's original subsumption architecture follows a horizontal layering approach [31]. An illustration of a horizontally-layered architecture is given in figure 6.

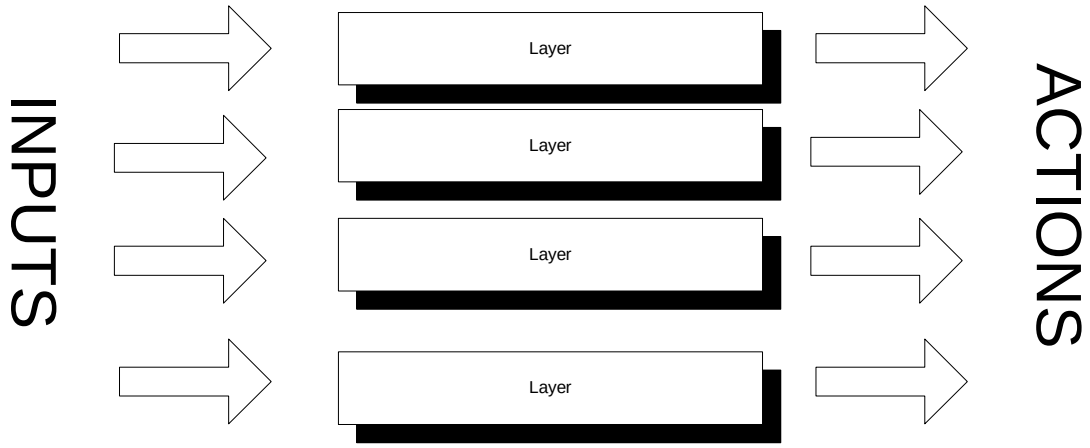

Figure 6. Horizontal Layering Agent Architecture

In the vertical layering approach layers are hierarchically ordered with the complexity of layers increasing with their level. Interaction between layers is defined as hierarchical. In other words, interaction between the bottom layer and the top layer cannot be direct. The interaction has to be done through intermediate layer(s), whereas in the case with horizontal layering, all layers execute in parallel. An illustration of a vertically layered architecture is given in figure 7.

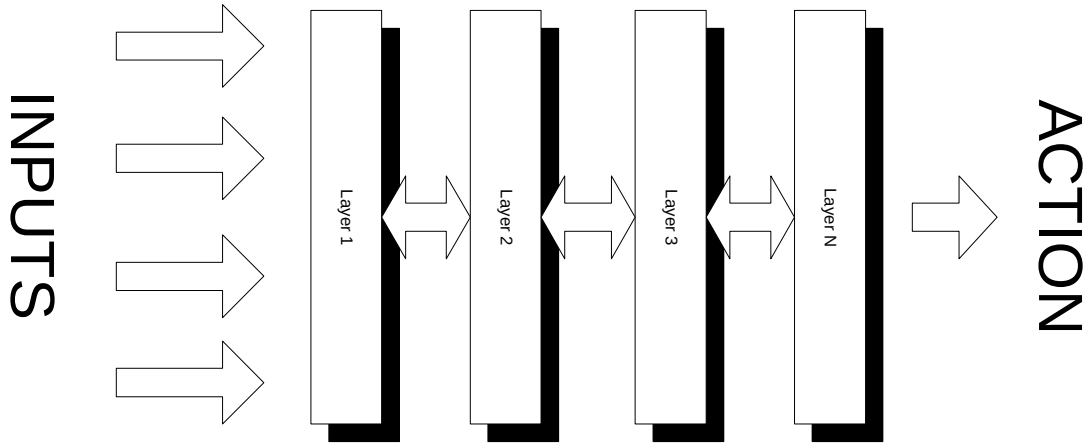

Figure 7. Vertical Layering Agent Architecture

Section 3.4.1 gave frequent reference to three-layered architectures. It seems, especially in the autonomous embodied agents field (robotics), that most researchers [22][73][173] have standardised on the use of three vertical layers. The reason for this is not so much theoretical, but based on a pragmatic approach and on the results of experimenting with various numbers of layers. Most hybrid agent architectures consist of a deliberative layer, a reactive layer and an interface between them. It is noted that, although the CLARAty architecture is a two-layer architecture, it is logically very similar to a three-layer architecture. An example of a typical three-layered architecture is given in figure 8 (modified from [22]).

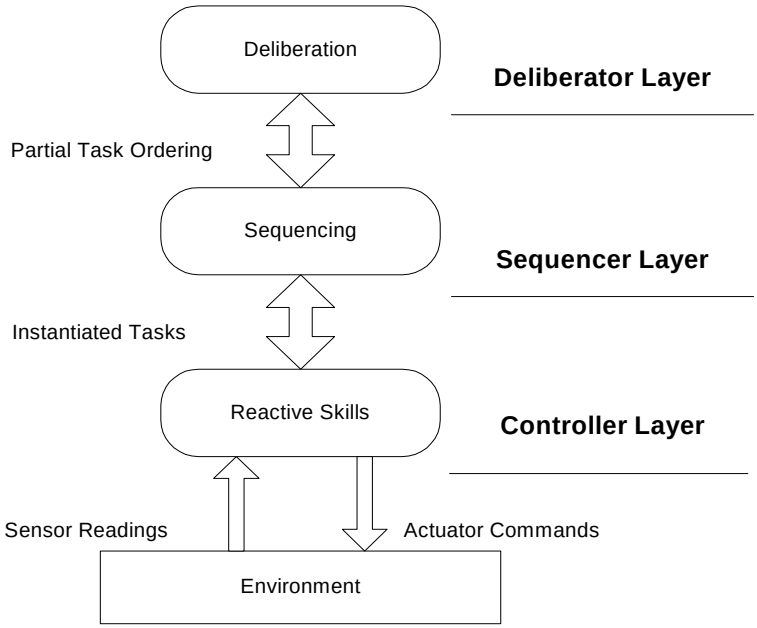

Figure 8. A Typical Three-Layer Agent Architecture

There are numerous, sometimes confusing naming conventions of these three layers. For the purpose of this thesis the adopted terminology is that of the ATLANTIS architecture [74]. The layers in the ATLANTIS architecture are called the controller, the sequencer and the deliberator layers. Each of these logical layers is described in greater detail in the next few sections. Naming conventions for the different architectures are summarised in Table 1.

Architecture	Top Layer	Middle Layer	Bottom Layer
ATLANTIS	Deliberator	Sequencer	Controller
3T	Planning Layer	Sequencing Layer	Skill Layer
TDL	Planning Layer	Executive Layer	Behaviour Layer

Table 1. Overview of Three-Layer Architecture Terminology

3.4.2.2 Controller Layer

The main purpose of the controller layer is to react dynamically, in real time, to changes in the environment. It can be seen as the implementation of fast feedback control loops, tightly coupling sensors to actuators [74]. The controller layer is usually implemented using a behaviour based robotics approach [109], as a set of behaviours. Behaviours (in behaviour based robotics terms) can also be seen as control laws that encapsulate sets of constraints in order to achieve specific behaviour [113].

Behaviours are usually implemented as handcrafted, simple, conditional rules. The behaviours take, as conditions, sensor readings and as action the behaviours provide motor actuation. The need to handcraft the behaviours is one of the challenges of behaviour based robotics that needs to be overcome. Because the behaviours interact with the real-world environment in real time, it is of crucial importance that these behaviours are fault-tolerant (or at least fault-aware) and that behaviours are bound by a maximum deliberation time. Behaviours are usually stateless; they do not maintain any local or global environmental models. Behaviours at this level are basic (primitive) behaviours. Basic behaviours are often combined into more complex behaviours by the next layer, the sequencer layer. Basic behaviours can be selected

either based on researchers' experience or according to a methodology such as that described in [113].

3.4.2.3 Sequencer Layer

The job of the sequencer layer is to manipulate basic behaviours into more complex behaviours that are closer to the symbolic layer, i.e. the deliberator. The sequencer layer achieves this task by enabling or disabling behaviours and/or by providing parameters for behaviours' execution. The storage mechanism for more complex behaviours is usually a library of plans for implementation that use basic behaviours. The task of breaking down a complex behaviour into basic behaviours is by no means trivial.

There are two main approaches to transform complex behaviours into basic behaviours:

- The universal plan approach, where all of the states and hard-coded complex behaviours are enumerated as a combination of basic behaviours, usually in table format [165].

- The conditional sequencing approach, where only the conditions that trigger behaviours are stored. Conditional sequencing can be implemented using either special, purpose-designed languages such as RAP [68] or as an extension of more traditional programming languages such as C, as was the case with TDL [173].

The sequencer layer can be seen as the interface between symbolic knowledge representation (as implemented in a deliberative layer) and sub-symbolic knowledge representation (as implemented in a controller layer).

3.4.2.4 Deliberator Layer

Knowledge representation in the deliberator layer is symbolic in nature. All reasoning is done using a symbolic world model. The symbolic reasoning approach and the

problems associated with this approach were discussed in section 3.2. The deliberator layer is the most abstract layer as it does not have direct interaction with the environment. However, the deliberator layer performs some of the crucial tasks in a hybrid agent architecture. The task of the deliberative layer is to perform the following functions:

- to build and maintains the world model,
- to deliberate (reason) on the course of action in symbolic terms, and
- to interface with the sequencer layer.

Being based on symbolic knowledge representation for its deliberation, the deliberator layer usually uses traditional artificial intelligence techniques such as planning and inference [197][72]. These techniques are traditionally computationally demanding and thus the deliberative layer does not respond to changes in the environment in real time. The controller, and to a lesser extent the sequencer layer, operate in real time. The deliberator expresses medium to long-term goals to the sequencer layer.

The deliberator layer interfaces with the sequencer layer either through plans that are presented to the sequencer layer, or responds to queries from the sequencer layer [74].

The deliberator layer is usually implemented in a standard, high level programming language, or using an inference engine (e.g. an expert system shell).

3.4.3 Hybrid Agent Architecture – 3T

In this section, an example of a hybrid agent architecture is presented and discussed in greater detail. The name 3T is derived from the three layers used by the architecture. An introduction to the 3T architecture is given in section 3.4.3.1, followed by an overview of the layers of the 3T architecture in sections 3.4.3.2 to 3.4.3.4. Section 3.4.3.5 provides a summary of the 3T architecture.

3.4.3.1 Introduction and an Overview

Most of the team members that created the 3T architecture [22] were involved with NASA and some of the 3T applications were related to space research programmes [22].

The 3T architecture was built upon various earlier research efforts [67][73] done by the same team, and designed with applications in embodied agents (robots) in mind. The three layers of the 3T architecture correspond to the layers that are described in sections 3.4.2.2 – 3.4.2.4.

For the purpose of the 3T overview, the original 3T terminology is used when describing the layers.

3.4.3.2 Skills Layer

The task of the skills layer, being the bottom layer of the 3T architecture, is to interact with the environment. A skill corresponds to a behaviour and its purpose is to achieve or maintain a particular state. Because the skills are dependant on the robot's physical implementation, any hard coding would seriously limit the architecture's flexibility. The approach was thus taken to implement a robot-independent skill representation based on work by Yu *et al* [202]. The representation consists of:

- Inputs and outputs, that are declarative descriptions of expected inputs and produced outputs. Outputs of one skill can be linked to inputs of another skill, thus allowing for chaining of skills.
- A computational transform, which forms the core of a skill, and which produces outputs according to computational rules, from inputs.
- An initialisation routine, that allows for skill initialisation in a secure and expected manner.
- An enable/disable function, that provides a sequencer with a mechanism to suppress and to enable a skill.

The enabled skills are executed in parallel. All skills interface with a skill manager that in turn interfaces with a sequencer, providing the sequencer with a single entry point to the skill layer. The skill manager handles all communications, asynchronous events handling and the enabling/disabling of skills.

3.4.3.3 Sequencing

The sequencing layer in 3T is implemented as a Reactive Action Packages (RAP) [67] interpreter. RAP is a LISP-based structure that is simply a description of the desired task to be achieved. It is important to note that a task is not unconditionally described in minute detail. The task description relies on the robot's perception of its environment. In other words, depending on the environmental perception (model), the task might be executed in a different manner. Each RAP has a sequence of skills that is either activated or deactivated in order to achieve an allocated task. This mechanism provides a sequencer to the skills layer communication mechanism. Specialised skills, called events, provide a communication channel from skills to sequencer layers. Events provide a feedback by communicating the perceived state of the environment. The sequencer layer uses this information to determine if a particular set of skills have been completed.

3.4.3.4 Planning

The sequencer layer of the 3T architecture does not perform optimised resource allocation nor does it organise the sequence of routine tasks that perform more complex and more useful tasks. These are the tasks of the deliberator layer. The planning layer operates on a higher level of abstraction than the sequencing layer. There are a few reasons for this level of abstraction. Firstly, a higher level of abstraction is more understandable by humans and the planning layer can often serve as a system-user interface. Secondly, a higher level of abstraction reduces the size of the search space.

The Adversarial Planner (AP) [60] is used in 3T. The planning takes the form of higher level RAP that are then decomposed into more elementary RAPs by the sequencer layer.

AP has two features that are considered very useful in robotic applications: it has the ability to control more than one agent at the same time and it can reason with agents that exhibit negative interaction (adversary attitude), for example agents that are not controlled by 3T (uncontrolled agents in 3T terminology).

Since the focus of this thesis is an architecture for cooperation between agents, the first feature is covered in more details. The multi-agent coordination mechanism in 3T allows for coordination between robots, but its usefulness is questionable. All coordination must be done through a central system. In other words, instead of having agents cooperate through consensus or some other coordination mechanism, it imposes a centralised, hierarchical control on otherwise autonomous agents. This severely restricts the potential of novel, self-organising multi-agent applications. Therefore, this thesis treats the 3T architecture as mainly a single-agent architecture.

3.4.3.5 3T – Conclusion

3T is a comprehensive architecture with some interesting features, for example the deliberator layer adversarial planning and its (albeit limited, as discussed in the previous section) provision for the coordination of multiple agents. Coupling between layers is coherent from the top to the bottom layers, but the upward flow of information is very limited.

The sequencing and deliberator layers are implemented in LISP. Although LISP has been one of the most commonly used programming languages in AI research, its applicability to real-world embodied agents is questionable. LISP has relatively large resource requirements (it is usually implemented as an interpreter) and its speed of execution is usually slower than that of compiled programming languages. Portability to different hardware, e.g. lower-end platforms (such as a cheap, swarm like robotic

system [185][184]) might also present a problem, due to the relatively high computational demands of a LISP interpreter.

3.5 Summary

The focus of this chapter was the application of various agent architectures to robotic applications. This chapter overviewed three types of agent architectures, namely symbolic, reactive and hybrid. These architectures were initially discussed in general terms. The general discussion was followed by a detailed review of representative examples of the three architectures.

The next chapter naturally follows this chapter by extending the discussion and overview to MAS architectures.

Chapter 4: Multi-Robot Architectures

Multi-robot architectures can be seen as a special case of multi-agent systems, where agents are embodied in their environment. This chapter focuses on multi-robot architectures specifically, since a more complete overview of MASs architectures is outside the scope of this thesis. Section 4.1 enumerates some of the early MASs together with a taxonomy for multi-robot teams. A behaviour based robotics approach to multi-robot teams is discussed in section 4.2. A hybrid multi-robot architecture, MACTA, is overviewed and discussed in section 4.3.

4.1 Introduction

In this section, some of the general MASs are overviewed. The first MAS architectures and related techniques have their origin in the Distributed Artificial Intelligence (DAI) field. The Contract Net Protocol (CNP) [49][175] can be considered as one of the first illustrations of the concept of agency. CNP is used to divide a task between multiple agents and it is a widely used protocol with many variations [161][46].

One of the first models to have actually implemented agents (albeit very simple agents) is ACTORS [1]. Since then many MASs have been implemented in fields as diverse as air traffic control [41], distributed vehicle monitoring [59] and more recently, Intrusion Detection Systems (IDS) [7]. As stated previously, the emphasis of this thesis is on embodied agents (robots) and from the next section, all classifications and examples are robotics-related. The reader is referred to [88][138] for more detail on general MASs.

With robotic MASs in mind, it is useful to consider a robotic MAS taxonomy. The chosen taxonomy for the purpose of this thesis is that of Dudek *et al* [56]. According to Dudek *et al* , robot teams are classified according to the following characteristics:

- Size of the team. Based on the size of robot teams, teams can be divided into classes ALONE, PAIR, LIM and INF, denoting one, two, multiple (but with a finite limited number of robots) and an unlimited number of robots respectively.
- Communication range. The teams are classified based on the communication range into NONE, NEAR and INF classes, denoting no communication, local communication (limited to the distance between robots) and unlimited distance communication (for example softbots, that can communicate using the Internet).
- Communication topology. According to the communication topology, multi-robot teams are classified as: BROAD, where a broadcasting mode of communication is used; ADD, where each agent has an address; TREE, where communication is done through a hierarchical network; and GRAPH, where communication links are defined as a graph.
- Communication bandwidth. Based on the cost of the communication, robot teams are divided into four classes: INF, where communication is free; MOTION, where the cost of communication is the same order of magnitude as the motion of the robot; LOW, where the cost of communication is greater than the cost of moving the robot; and ZERO, which indicates that no communication is possible.
- Collective reconfigurability. The collective reconfigurability indicates the rate at which the robot team can re-organise itself spatially. It divides the robot collective into three distinct classes: STATIC, where the topology is fixed; COMM, where members coordinate to achieve the reorganisation task; and DYN, where the spatial relationship can change arbitrarily.
- Processing ability of each collective unit. Robot teams are divided into classes based on the computational model of each robot. The view adopted in this thesis is that classification based on computational model is too limited, as it does not take in consideration the real processing power or complexities of the adopted world model representation (if any). However, for the sake of completeness, the four classes are enumerated next: SUM, where processing power is equivalent to that of a non-linear summation unit (i.e. a neuron in artificial neural networks); FSA, where processing power is equivalent to a finite state automaton; PDA, where processing power is equivalent to a push down automaton and TME, where processing power is that of a Turing machine equivalent. For more detail, the reader is referred to [56].

- Collective composition. A robot team can be classified according to the physical attributes of robots: IDENT, where all agents are identical; HOM, which indicates homogenous MAS where agents essentially have the same characteristics; and HET, where agents are heterogeneous with different physical characteristics. For a formal approach to measuring robot group diversity, reader is referred to [12][14].

In the remainder of this chapter, two multi-robot architectures, the Behaviour Based Robotic (BBR) and Multiple Automata for Complex Task Achievement (MACTA) are overviewed and discussed. Note that the BBR has evolved over time, but for the purpose of this thesis the first, original, version as described in [113] is considered.

Using the adopted taxonomy, these two architectures can be described as given in table 2.

Robot Team Characteristic	BBR	MACTA
Size of Team	LIM	PAIR
Communication Range	NEAR	NONE
Communication Topology	BROAD	Not Applicable
Communication Bandwidth	MOTION	Not Apllicable
Collective Reconfigurability	DYN	STATIC
Processing Power of a Team Member	TME	TME
Collective Composition	HET/HOM	HET/HOM

Table 2 Comparison of BBR and MACTA architectures

4.2 Behaviour Based Robotics

There were various efforts to improve on reactive architectures capabilities. Some of these effort were mentioned in section 3.3.3. The behaviour based robotics architecture, as developed by an MIT team headed by Matarić [109], is discussed to illustrate the ideas behind a behaviour based approach for MAS.

4.2.1 Introduction

Behaviour based robotics can be seen as an extension of purely reactive architectures [108]. Behaviour based robotics has the concept "behaviour" as its foundation. In addition to the definition of behaviour as given in section 3.4.2.2, a behaviour can also be defined as a piece of code that produces behaviour when it executes [74].

BBR provides mechanisms for cooperation, coordination, communication and planning [109]. In general, BBR are decentralised systems of autonomous agents. Cao *et al* [38] consider BBR to be a swarm-like architecture. This view can be accepted to a degree, but BBR is certainly more advanced than a classical swarm system such as ANT [119N, 181] as it allows for explicit communication and learning.

BBR adopts a building block approach, similar to that used by the subsumption architecture (see section 3.3.3), that relies on developing basic behaviours first. Once basic behaviours are thoroughly tested, the basic behaviours are then combined into more complex behaviours.

Behaviour based agents can become quite complex when basic behaviours are combined in more complex ways, or when the side effects of its behaviour and interaction with the environment can yield some useful characteristics, such as emerging behaviour [178].

The concept of intelligent emerging behaviour is a paramount premise of reactive and behaviour based architectures. The belief that intelligent behaviour will emerge through the interaction of a system with its environment is based on the observation of natural systems, for example ant colonies [178]. However, there is still no formalisation of a process that will allow new intelligent behaviours to emerge in multi-robot teams. In this thesis, proposed emergent behaviour is exhibited on a higher level of abstraction – the social level, thus the topic of emergent low-level behaviours through interaction with an environment falls outside the scope of this thesis.

BBR uses a decentralised approach where all robots are fully autonomous with sparse communication between robots. Furthermore, knowledge representation is not symbolic. Hence there is no high level planner nor any symbolic learning mechanism.

4.2.2 Basic Behaviours

The building block approach of BBR assumes that simplistic basic behaviours can be combined into more complex ones at a higher level of abstraction. BBR also proposes a methodology for choosing such basic behaviours. That being said, the process of choosing basic behaviours is still a heuristic process, without a fixed metric for selecting an "optimal" set of basic behaviours. Matarić proposes two criteria for defining the set of basic behaviours [113]:

- *Necessity*. The set of basic behaviours must contain only behaviours that are necessary to achieve the desired goals in a given domain.
- *Sufficiency*. The set of basic behaviours must contain behaviours that will be sufficient to achieve the desired goal in a given domain.

Benchmark tasks for robot teams that must be achieved by combining basic behaviours are foraging, flocking, herding and surrounding. To achieve these tasks (assuming a two-dimensional spatial domain) the following basic behaviours have been identified [110][113]:

- *Safe wandering*, which refers to the ability of a robot team to move around while avoiding obstacles and collisions with other agents. This behaviour is the basis for any exploration or mapping task.
- *Following*, which refers the ability of an agent to follow another agent (the leader) and maintaining a following distance from the leader.
- *Dispersion*, which refers to the ability to spread out and to maintain a minimum distance between agents.
- *Aggregation*, which is the opposite of dispersion, i.e. the ability to gather and maintain a maximum distance between the agents.

- Homing, which refers to the ability to find a particular region or location. This ability is necessary for tasks such as sample collection, foraging and self-preservation.

BBR has been designed as an architecture that can be easily extended by implementing more complex behaviours. This characteristic is desired and required for an emergent behaviour approach.

The strength of a behaviour based architecture also lies in the fact that incorporation into more complex hybrid architectures is easy without any significant modifications. The behaviour based architecture is used in some well known hybrid MAS such as MACTA [11][10] and InterRap [129].

4.2.3 Learning in BBR

The goal of any learning mechanism is to optimise system performance. There are various applications of learning mechanisms that can improve the performance of the system [91]. Applied to BBR, learning can be divided into learning new behaviours, learning new facts about the environment and learning to improve behaviours. It is important to note that learning in BBR is not based on symbolic reasoning. In BBR, learning uses behaviours that are in their nature sub-symbolic structures.

There has been significant improvement in the learning capabilities of BBR MASs over the past few years. Initially, research focused on learning behaviour policies, in other words on improving the behaviour selection process through learning [113]. More recent research has complemented the behaviour policies learning with learning from environment models and from interaction with other agents [115]. An overview of learning in BBR is given next.

4.2.3.1 Learning Behaviour Policies

The behaviour policies determine which behaviours are selected for execution. The goal of learning behaviour policies is to improve on the selection of appropriate

behaviours for specific tasks and environment conditions. Reinforcement learning [91] is one of the most frequently used mechanisms in AI. In BBR, the reinforcement learning is used to learn behaviour policies. Reinforcement learning relies heavily on stimuli-response coupling, and it has been used in robotics, for example in [25].

Most reinforcement learning models that were successfully applied to computational learning are based on Markov Decision Process (MDP) models [113]. Unfortunately MDPs, when applied to embodied, situated agents, assumes that interaction between an agent and its environment can be viewed as synchronised finite-state automata that are deterministic and predictable. This is not always the case, due to uncertainty in sensing of the environment. For MDP to be applicable to learning of behaviour policies, the following modifications have been made [113]. These modifications are described next.

- In robotic applications of MDP, the state space defines sensor inputs of the robot, together with the set of internal parameters of the robot. The first modification is that the state space has been defined by conditions instead of full state descriptions. The space of conditions is generally much smaller than the complete state space and allows for faster computations.

- The reward function is focused on achieving pre-determined sub-goals of concurrent behaviour. This is a very important departure from the original MDP model that prefers a sequential process. At any moment, in BBR robotic applications there are many active behaviours, contributing by their actions towards a completion of a high-level goal. Concurrent reward functions can then focus on improving each behaviour. The improvement in behaviours leads to the improvement of overall system performance.

- A progress estimator function is introduced as a part of the learning mechanism. The role of the progress estimator function is dual: instead of rewarding only after a specific goal has been achieved, an intermittent reward allocation is used. The progress estimator function uses a mechanism for a termination of a specific behaviour [113].

The MDP was further modified, by using reward sharing mechanism that punishes greedy behaviour of agents [116][118].

4.2.3.2 Learning Environment Model

One of the main characteristics of BBR is a decentralised approach and a lack of symbolic representation of its environment. Modelling of the environment in BBR is reduced to creating a world map, based on exploration. Even learning of such a simplified environmental model presents a challenge to BBR architectures. The problem of learning a world map was solved through the mapping based on location of landmarks [112]. In BBR there is no provision for storing traditional symbolic knowledge that can describe a world map. Instead world map information is stored by creating behaviours that store landmark information, as described next.

The mapping based on location of landmarks proposes that once a landmark is detected, a new behaviour is created. The newly created behaviour has the following parameters: landmark type, direction (orientation) and distance from the previous landmark. Each new behaviour is linked to the previous one, in effect creating a map that consists of a set of linked landmarks. Navigation then consists of traversing from one landmark to another.

More details about the landmark navigation approach to mapping can be found in [112][50].

4.2.3.3 Learning Behaviour Patterns from Behaviour History

The BBR architecture incorporates a mechanism to learn behaviour patterns from the history of behaviour dynamics [124]. Each robot maintains a tree-like topological structure for the duration of task execution, wherein occurred behaviours are stored.

The tree topology represents the robot's behaviour space, where each node represents an executed behaviour. The tree topology is the representation of a behaviour

sequence in a robot's behaviour space. The links were weighted statistically, according to link-usage frequency.

For the different goals, patterns in behaviour activations are identified using the tree of behaviours. If a pattern in behaviours activation has led to the fulfilment of the identified goal, the identified pattern is used in subsequent behaviour selection of similar tasks. The approach was successfully tested on robot systems. More details on these experiments can be found in [124].

4.2.3.4 Other Learning Methods in BBR

Various other learning methods were tested in the BBR architecture, some using innovative approaches. For example, to facilitate learning of interaction models between agents, the authors have developed Augmented Makarov Models (AMM) [119]. In order to provide a higher level of abstraction, necessary for more complex deliberation, an abstract behaviour was introduced in [135] that have provided BBR with reasoning tools almost equal to those of the best deliberative, symbolic systems.

4.2.4 Cooperation Model

Cooperation does exist in BBR but it is not obvious, as there is no intentional cooperation. Instead, cooperative behaviour can emerge as a side-effect. At this stage it is important to note a view firstly expressed by Matarić [114] that cooperative behaviour based on negotiations require direct communication between the agents. Communication is also a prerequisite for distributed cooperative problem solving [80]. With these views in mind, the cooperation model used in BBR has limited capabilities, mostly because of the limitations of the communication method.

Firstly, BBR is a highly decentralised architecture with fully autonomous agents. There is no central system or knowledge repository that can store the data that was acquired by multiple agents. Instead, each agent is dependant on its local environment for the knowledge it acquires. In other words, the knowledge is not shared.

Secondly, the communication method is capable of broadcasting only short messages. This further limits the amount of information that can be transferred between the agents (note that the communication protocol transmits only simple messages, not knowledge).

Thirdly, there is no symbolic knowledge information. The symbolic knowledge is easy to exchange and modify, using some of the more advanced approaches such as KQML [66][99] or the XML [186].

Initially, communication was used for the simple task of detecting another robot [113]. At a later stage, a robot could "agree" on team interaction based on information received from another agent [119]. Limited social behaviour was modelled using the same, limited communication mechanisms [117]. Despite the above-mentioned limitations, cooperation was investigated in [174] with a limited success.

Social behaviour is a way of resolving conflicts that arise in teams of autonomous, uncoordinated robots. Conflicts in BBR do not arise because of the competitive nature of agents but due to the unintentional interference between the agents. Agents do not have conflicting goals but the interference may lead to negative interaction.

Out of the three coordination mechanisms that were discussed in section 2.4.1.1, the BBR architecture was modified to use social coordination. Social coordination is the predominant approach to coordination in BBR [117][116].

4.2.5 BBR - Conclusion

As a successor to purely reactive architectures, BBR has continued the tradition of revolutionising the field of mobile embodied agents.

BBR has retained all positive characteristics of reactive architectures. BBR is characterised by fast execution time, due to simplistic couplings between sensors and actuators. BBR also lends itself to relatively simple robot implementations that provide an efficient environment for multiple robot team experiments.

BBR has significantly improved over purely reactive architectures by providing mechanisms for the internal representation of the environment [50], learning and improving on existing behaviours [118], communication [119] and even social behaviours [117].

One of the main characteristics of BBR is its limited communication model, which may seem to be a major limitation. However, considering the limited knowledge that is maintained by each agent, the limitations of its communication model are not so severe.

Cooperation, internal representation of the environment and even social behaviours are implemented in a very efficient manner, but all of them have limitations, as described in previous sections.

It is interesting to note that BBR has evolved from reactive architecture characteristics toward characteristics that are traditionally associated with symbolic architectures (e.g. planning, social interaction and environment modelling).

One of the main reasons for the appearance of reactive architectures and subsequent improvements of the reactive architecture (such as BBR) was the fact that the execution of traditional symbolic reasoning based systems was slow. Today, processing power is easily and cheaply available and this approach should be reconsidered. It seems that the future of behaviour based systems lies in its incorporation with hybrid systems as an environment-agent layer.

4.3 Hybrid MAS (MACTA)

This section presents an overview of a hybrid architecture, Multiple Automata for Complex Task Achievement (MACTA), developed at the University of Salford [11][10].

4.3.1 Introduction

MACTA combines a symbolic planner with a behavioural architecture, the Behavioural System Architecture (BSA) [17][15]. MACTA is referred to as a MAS with a reflective agent that supervises multiple behaviour architecture agents (robots) [11]. MACTA is not only a hybrid in the sense that it combines sub-symbolic and symbolic reasoning, but it also consists of embedded agents (robots) and a non-embedded agent (reflective agent). MACTA has been successfully tested on tasks such as docking, cooperative relocation of objects and tracking using two robots. MACTA consists of two main components, namely BSA and the reflective agent.

Although it consists of two main components, MACTA is a three-layer hybrid architecture because it also contains an interface layer, referred to as the mission organiser.

4.3.2 Behavioural Synthesis Architecture

BSA [17][15] extends and improves earlier reactive architectures such as the subsumption architecture [31]. BSA allows for single and multiple robot systems. Within BSA, two types of robotic behaviour are identified:

- Egoistic, or self-interested behaviour where a robot pursues only its goals, without taking into consideration the goal of the whole multi-robot team.
- Altruistic, where a robot might not pursue its optimal state, in order to allow the whole multi-robot team to achieve its optimal state.

Egoistic (or self interested) behaviour is a typical characteristic in a single robot system. Altruistic behaviour is often desired in case of multi-robot systems.

BSA defines four layers, namely the self, the environment, the species and the universe layers (refer to figure 9):

- The self-layer contains strategies concerned with internal resources (e.g. battery levels).
- The environment layer contains behaviours that relate to the agent's interaction with the environment (such as obstacle avoidance).
- The species layer contains strategies for the interaction between agents (such as cooperation).
- The last layer, the universe layer, contains strategies for achieving tasks (such as object reallocation).

BSA agents have no representation of the world. Instead, the BSA agents are aware only of their local environment. BSA, in the tradition of many reactive and behavioural architectures, is horizontally layered, where each layer can receive inputs from the environment and each layer can produce actuator commands.

Where BSA departs from reactive architectures is that the actuator command is synthesised from all actuator commands and only the resulting command is sent to the physical actuator. This process is illustrated in figure 9. The actuator commands produced from a single behaviour are represented in a form of vectors in two-dimensional space. The synthesised output, represented as a vector on the right hand side of figure 9, is then the result of summing all individual vectors.

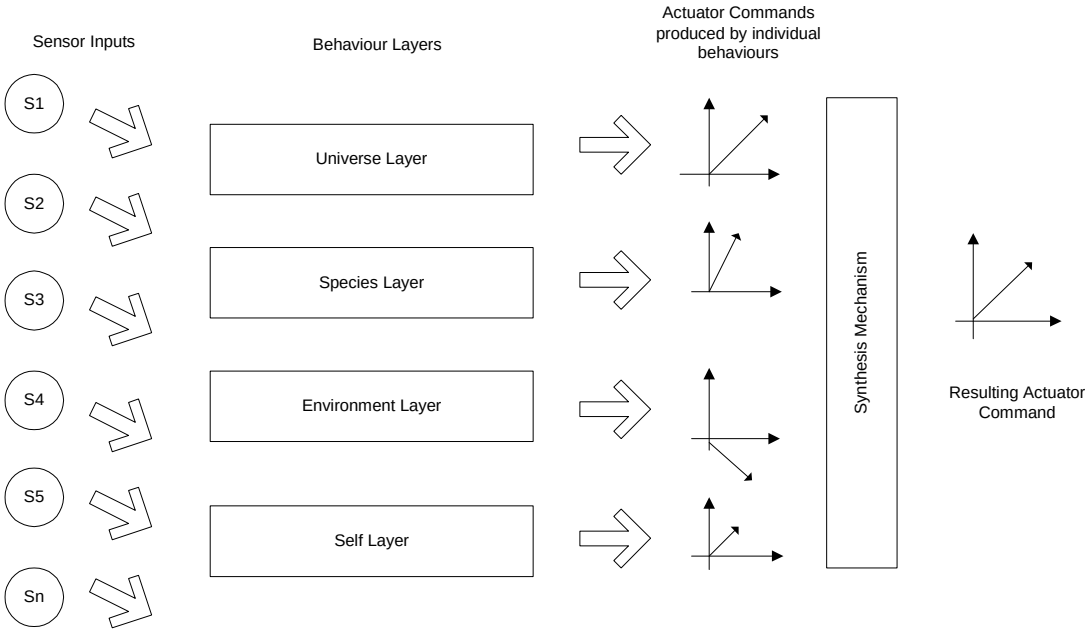

Figure 9. BSA Architecture illustrated (modified from [16]).

The synthesis mechanism is not the only coordination mechanism available to BSA. For example, coordination can be achieved by creating complex behaviours scripts with predetermined coordination rules. The complex behaviours can enable and disable simpler behaviours according to these coordination rules, thus preventing conflicting interaction between the behaviours.

4.3.3 Behaviour Scripts

Behaviour scripts serve as an interface between the behavioural architecture and the reflective agent. Behaviour scripts in BSA are in the form of a triplet (*sensor-pre-conditions, set of enabled behaviours, sensor post-conditions*).

The behaviour script can be seen as a set of enabled behaviours that are activated when sensor pre-condition is satisfied and remains active until sensor post-conditions are satisfied. It is important to note that scripts do not contain behaviours. The behaviours are contained in the BSA and organised in four layers, as explained in the previous section. The behaviours are activated or deactivated by appropriately setting the *active flag* associated with every behaviour, irrespective of the layer in which it is stored. The actuator output is then synthesised from the outputs of all active behaviours.

The process is illustrated below in Algorithm 1 (from [9]):

```
If (sensor-pre-conditions TRUE) then

        While NOT (sensor-post-condition)
                Synthesise(activebehaviours)
        Endwhile

endif
```

Algorithm 1. BSA behavior algorithm

Behaviour scripts correspond to the sequencer layer of hybrid architectures (see section 3.4.2.3). The original MACTA implementation used handcrafted behaviour scripts [11][16].

4.3.4 Reflective Agent

MACTA defines a fifth layer, the reflective agent, to maintain the symbolic world model and to perform symbolic reasoning. Although, logically, the reflective agent forms another layer of the MACTA architecture, the MACTA terminology refers to it as a reflective agent. In order to avoid possible confusion, the MACTA terminology is used from now onwards. The reflective agent has two main components: the mission organiser and the planner, both of which are described in this section. The planner creates a partial order plan (consisting of numerous sub-goals), using the symbolic reasoning mechanism, which is then passed to the mission organiser. The mission organiser then translates the symbolic sub-goals into behaviour scripts that are, in turn, passed to the agents (robots). A high level overview of the reflective agent is illustrated in a figure 10.

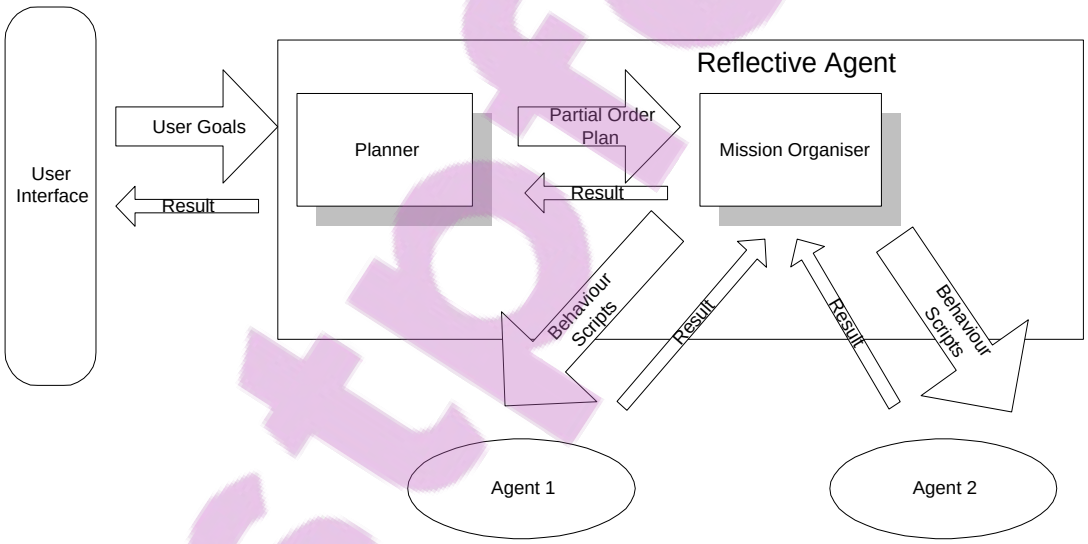

Figure 10. Overview of MACTA Reflective Agent

Interaction of a reflective agent with BSA agents is via behaviour scripts. The feedback from BSA agents to a reflective agent is very crude: the feedback is reduced to an indication as to whether the execution of a task has been successful or not. The planner and mission organiser are discussed next.

4.3.4.1 Planner

MACTA uses a standard symbolic planner [9], namely a modified UCPOP [146]. The UCPOP is modified as follows:

- Provision is made for multi-robot teams. Actions are, however, manually handcrafted.

- Task allocation is done in a simple way that uses all the available robots for a task. While this approach worked well for the experiments using the MACTA architecture (where only two robots have been used), it is unclear how (and if) this approach can be scaled up to robot teams consisting of a larger number of robots. An alternative task allocation mechanism using a market inspired approach, such as CNP [49], was considered but not implemented.

- The planner does not maintain a full world model. It contains only pre-programmed information about stationary objects, for example, walls.

The planner subdivides goals into a partially ordered plan that cannot be decomposed further. The partially ordered plan is then presented to the mission organiser.

4.3.4.2 Mission Organiser

The mission organiser's responsibility is to translate the symbolic representation of the user's desired goal (obtained via an user interface) and pass it to the planner. The planner then generates a hierarchical non-linear plan. The mission organiser translates the primitive (but still represented in symbolic terms) actions of a plan into behaviour scripts (that are represented in sub-symbolic terms), which are then propagated to appropriate agents. The role of the mission organiser is illustrated in figure 11.

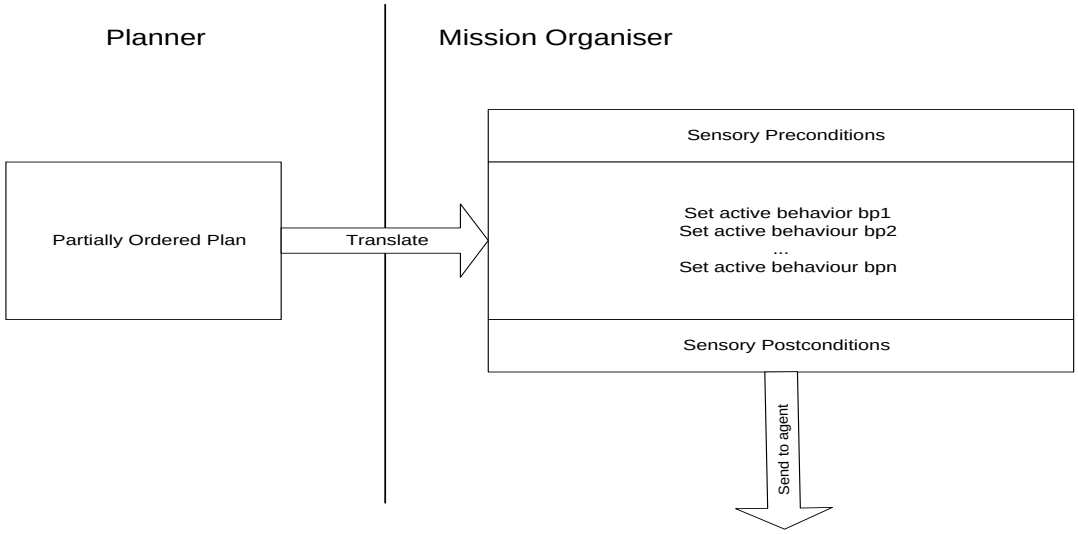

Figure 11. Role of the Mission Organiser in MACTA (adapted from [10])

4.3.5 Coordination Model

MACTA caters for coordination between agents in a very elementary fashion. Coordination is achieved through behaviours only, without exchanging any information and without maintaining a world model. In the experiments that were performed, MACTA was successfully applied to tasks such as cooperative object relocation and cooperative object tracking [11][10]. For each task, the reflective agent initiated coordination.

There are two modes of interaction between robots: interaction between close-coupled robots and interaction between loose-coupled robots. In MACTA, a close-coupled scenario assumes physical coupling, i.e. robots are physically attached to each other. In loose-coupling, robots are reliant on remote sensing equipment such as cameras.

The coordination model used in MACTA has numerous shortcomings. Firstly, the reflective agent is always the initiator of coordination behaviour. This is not ideal in a highly distributed environment where agents are complex because resource usage is inefficient. Secondly, all behaviours for coordination are handcrafted. It is improbable that handcrafted solutions can be scaled to large robotic teams. Thirdly, interaction between robots is done purely through sensing of the local environment. This is a characteristic of reactive and behaviour based architectures and a critique of these was presented in section 3.3.4.

In MACTA, although agents are autonomous, the coordination model used indicates that all planning is done centrally. MACTA can therefore be seen as a centralised architecture.

4.3.6 MACTA - Conclusion

The MACTA architecture is a promising hybrid architecture that combines the best characteristics of a symbolic planner with an efficient behaviour architecture. However, there are some aspects that need improvement. For example, all behaviour scripts are handcrafted, which is clearly not feasible for a large number of heterogeneous agents. The mission organiser component of the system can also be improved. The current system uses one-to-one mapping of a primitive action (the output of a planner component) to a behaviour script. Intuitively, an one-to-many mapping would potentially be more powerful (and abstract), but it would introduce additional complexity. In other words, a desired task could be achieved through different behaviours, depending on the conditions and inputs from other agents or the environment.

MACTA does not provide any learning mechanisms, which is another serious shortcoming. However, the primary problem with MACTA is that the majority of the model is either handcrafted or it caters for a team of only two robots, without a clear indication of how it could scale to larger teams of robots.

4.4 Summary

Two MAS architectures, together with a representative example, were overviewed and discussed in this chapter. Particular consideration was given to the coordination model employed by both architectures, as the coordination model is seen as the key to successful MASs.

The next chapter presents a new MAS architecture, INDABA, which addresses one of the most important aspects of MAS architectures, namely the ease of implementation of the coordination mechanism.

Chapter 5: New INtelligent Distributed Agent Based Architecture

As seen in chapters 3 and 4, agent architectures are almost as diverse as agent applications. Currently there is no architecture that will suit all applications. The new proposed INtelligent Distributed Agent Based Architecture (INDABA) is designed with the goal of constructing an architecture for cooperative, embodied agents (robots) in multi-robot teams. The architecture presented in this chapter is mainly a conceptual framework that is not too prescriptive in implementation technique. Instead, INDABA should be seen as a guideline for designing cooperative agents. An overview of INDABA and rationale behind INDABA are given in section 5.1. Section 5.2 presents the first layer of INDABA, the controller layer, together with an example that illustrates a potential implementation. The second INDABA layer, the sequencer layer, is presented in section 5.3, again together with an example that illustrates a potential implementation. Section 5.4 presents the concept of deliberator layer with an example that illustrates its workings. The last INDABA layer, the interaction layer, is presented in section 5.5. The example used to illustrate the interaction layer and associated concepts is based on a cooperative problem-solving approach that consists of five steps, as described in section 5.5. Section 5.6 summarises INDABA and outlines possible future developments.

5.1 Overview of INDABA

Based on the investigations of agent architectures that were presented in chapters 3 and 4, the following was observed:

- The hybrid architectures have clear advantages over pure symbolic and reactive architectures as they have the best characteristics of both approaches and they address the weaknesses of both approaches.
- Due to the various agents' (and robots' in particular) technology platforms, as well as the numerous possible applications, it is not possible to have a unified, general-purpose architecture that will satisfy all requirements.

- The coordination mechanism in the majority of existing architectures is not flexible enough or virtually non-existent.
- The coordination mechanisms often ignore uncertainty about a task by assuming the ideal environment where details about the task are complete and accurate.

With these findings in mind, the new proposed conceptual architecture, INDABA, was designed and developed. INDABA is a layered architecture. It appears that most researchers, i.e. Brooks and Barnes [31][15] agree that agent architecture should be layered [106]. Furthermore, in the field of autonomous robots, it seems that most of the researchers [22][73][173] have standardised on hybrid architectures, consisting of three vertical layers.

As discussed in section 3.4.2.1, architectures can either be vertically or horizontally layered. INDABA provides for a hybrid between these two approaches, albeit more of a vertical layering approach than horizontal. The layers of INDABA are illustrated in figure 12. In INDABA, the main interaction between the layers is between the vertical layers, while less frequent interaction between the agents is done through the horizontal layers. The lack of interaction between layers was one of the main critiques of many horizontal-layering architectures, such as the subsumption [31] and behaviour based architectures [113].

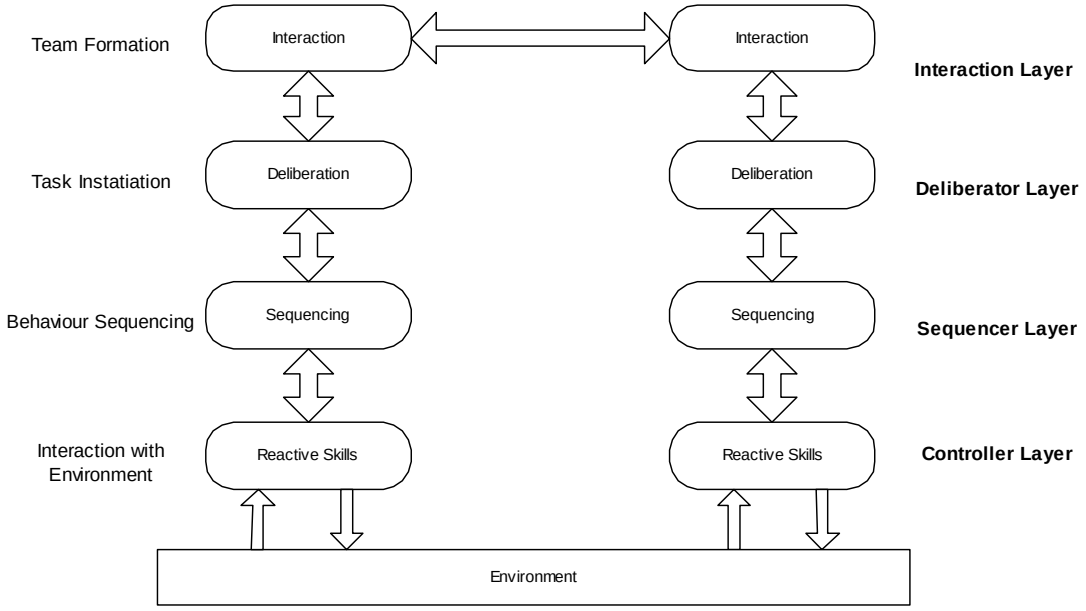

Figure 12. INDABA Layers

In comparison with more common three-layer architectures such as 3T [22] and ATLANTIS [73], INDABA introduces an additional layer; the interaction layer, that facilitates coordination through task allocation.

INDABA is designed with ease of coordination between the agents in mind. Ease of coordination between agents was achieved through introduction of a coordination-oriented layer that encapsulates the selected coordination mechanism.

INDABA is also a hybrid architecture. Currently, the most common approach to design robots is to use a hybrid approach (as described in 3.4) that combines the best characteristics of reactive and symbolic architectures (see sections 3.2 and 3.3 respectively). As indicated in table 1, different architectures use different layer naming conventions. INDABA adopts the layer names deliberator, sequencer, controller and interaction layers.

Each of the INDABA layers is discussed next, starting with the lowest level layer, i.e. the controller layer.

5.2 Controller Layer

The main purpose of the controller layer is to react dynamically, in real time, to changes in the environment. The controller layer can be seen as the implementation of fast feedback control loops, tightly coupling sensors to actuators [65]. In hybrid architectures [22][73], the controller layer is usually implemented as a set of behaviours using a behaviour based approach.

INDABA also implements the controller layer using behaviours. Behaviours implemented in the controller layer are basic (primitive) behaviours. Basic behaviours can be combined into more complex behaviours. Basic behaviours can be selected either according to the researcher's experience or according to a methodology such as described in [113].

Implementation of the behaviours in INDABA follows the guidelines given by Matarić [113]. Generation of simple behaviours such as *move_forward*, *turn_left*, *turn_right* and other simple behaviours, precedes the synthesis of more complex ones such as *avoid_obstacle*, *go_to* etc.

An example of a simple behaviour, *move_forward*, is given in algorithm 2 for a simple differential drive robotic platform, such as Lego Mindstorm [185].

```
behaviour move_forward
  while true
    LeftMotor(On)
    RightMotor(On)
  end while
end behaviour
```

Algorithm 2. move_forward behavior

Most of the simple behaviours have no limit on duration of their execution. As such, simple behaviours are controlled by more complex behaviours. More complex behaviours can start and stop simpler behaviours.

Complex behaviours usually have an associated completion condition. If the completion condition is satisfied, then the complex behaviour terminates. Alternatively, a complex behaviour might have a resource restraint that governs its execution. For example, a complex behaviour can be allowed to execute only for a limited period of time. In other words, behaviours can stop their own execution if stopping criteria are defined and the stopping criteria are met.

To illustrate a complex behaviour, consider an implementation of the *avoid_obstacle* behaviour. The *avoid_obstacle* behaviour in this example is activated if touch sensors detect an obstacle. The value for the wait function has been arbitrarily set to 50 (the parameter value indicates the hundredth part of a second, so a value of 50 indicates 0.5 seconds). The *avoid_obstacle* behaviour has a limited duration of its execution. The behaviour is implemented as a combination of simpler behaviours *move_forward,*

move_backward, turn_right, turn_left, detect_left_touch and *detect_right_touch*, as illustrated in algorithm 3.

```
behaviour avoid_obstacle
  if detect_left_touch
     start move_backward
     wait (50)
     stop move_backward
     start turn_right
     wait(50)
     stop turn_right
  else
     if detect_right_touch
       start move_backward
       wait (50)
       stop move_backward
       start turn_left
       wait(50)
       stop turn_left
   end else if
end behaviour
```

Algorithm 3. avoid_obstacle behaviour

It is important to note that INDABA does not prescribe specific implementations of behaviour. Implementations usually depend on the robotic platform, and INDABA's goal is to provide a platform independent architecture. Each behaviour is treated as a black box and as an autonomous, self-contained object. Using such approach, a multitude of various platforms, some of them with existing comprehensive libraries of behaviours, can be easily encapsulated into an INDABA agent architecture.

While higher layers can be interchangeable between various hardware and software platforms, the controller layer is platform-dependent, because it executes on a specific robot platform. The implementation of the controller depends on a physical suite of

sensors and actuators. Chapters 7, 8 and 9 discuss specific implementations of INDABA. The first two implementations runs as a set of algorithms in a simulated robot environment, and the third as a set of behaviours executing on a robotic platform.

5.3 Sequencer Layer

The job of the sequencer layer is to further combine behaviours into more complex behaviours that are closer to higher level goals. The sequencer layer achieves this task by enabling or disabling behaviours and/or by providing parameters for the execution of behaviours. The complex behaviours in the sequencer layer are seen as sub-tasks, used by a symbolic reasoning mechanism implemented in the next layer, the deliberator layer.

As discussed in section 3.4.2.3, there are different ways in which the sequencer layer can be implemented. For the initial implementation of INDABA, the universal plan approach [165] was adopted. The sequencer layer is based on a universal plan in the form of a table that is loaded from a text file. Each sub-task has a set of corresponding active behaviours and a condition or set of conditions that will satisfy its goal. The conditions are usually represented as a combination of completions of simpler behaviours.

Table 3 illustrates an implementation of a sequencer layer. Behaviours *safe_wander*, *detect*, *collect* and *home* are complex behaviours that are implemented in the controller layer, while sequencer layer behaviours ***Find***, ***Collect*** and ***Home*** are implemented as a combination of these complex behaviours in the controller layer (refer to table 3).

Sub-task	Active Behaviours				Goal
	safe_wander	*detect*	*collect*	*home*	
Find	1	1	0	0	*detect*
Collect	0	1	0	0	*collect*
Home	0	0	0	1	*home*

Table 3. Illustration of INDABA sequencer layer

For the example illustrated in table 2, it is important to note that the *detect*, *collect* and *home* behaviours do have completion conditions, while *safe_wander* does not.

The sequencer layer can be seen as a higher abstraction of basic behaviours. The result is a group of sub-tasks that can now be instantiated from the deliberator layer.

5.4 Deliberator Layer

The next layer of INDABA is the deliberator layer. The deliberator layer is the first INDABA layer that uses symbolic reasoning based on a symbolic world model. However, the deliberator layer performs crucial functions in INDABA (as for any other hybrid agent architectures):

- Builds and maintains the world model.
- Deliberates (reasons) on a course of action in symbolic terms.
- Interfaces with the sequencer layer, by starting s in the sequencer layer.

The initial INDABA implementation uses a simple backward chaining inference engine and a rule database to implement the deliberator layer.

To illustrate the working of the deliberator layer, consider a simple foraging problem. For this purpose, the sufficient set of rules are:

Task: **FORAGE**
Rule1: IF ***Find*** THEN ***Collect***
Rule2: IF ***Collect*** THEN ***Home***

The rules are loaded from a text file. The process start by pursuing the goal ***Home***. Simple backward chaining leads to goal ***Find*** .

Goal ***Find*** is then sent to the sequencer layer. The sequencer layer then performs a table look-up to determine the active behaviours associated with goal ***Find*** and the stopping criteria. From table 2, the active behaviours for goal ***Find*** are *safe_wander*

and *detect*. The stopping criterion for goal ***Find*** is that the behaviour *detect* is completed.

If the *detect* behaviour is completed, an object is detected and the sequencer layer reports to the deliberator layer that it has achieved its given goal, i.e. ***Find***. The deliberator layer then, from Rule 1, inferences that the ***Collect*** behaviour needs to be satisfied. The ***Collect*** goal is then passed to the sequencer layer, and the execution continues.

The above example is simple, but sufficient to describe basic execution of a task in INDABA. It is important to note that in the implementation of INDABA, as presented in this thesis, the deliberator layer does not build its own world model.

5.5 Interaction Layer

The interaction layer encapsulates mechanisms that facilitate the coordination between agents in INDABA. The interaction layer maintains its own internal state by means of maintaining multiple variables. The variables in INDABA are divided into sets that are referred to as mental states. In INDABA, each agent maintains its own set of mental states. There are three separate sub-sets of mental states, namely: self-related, task-related and society-related mental states.

Mental states can be changed by the agent itself, based on the agent's own experience, or they can be changed through interaction with other agents. Each of the mental states implemented in INDABA are described next.

5.5.1 Self-Related Mental State

The self-related sub-set of mental states consists of the agent's beliefs about its own capabilities. The initial application of INDABA is in robotics, where this sub-set consists of a robot's enumeration of its own sensors and actuators and their characteristics. In INDABA, the self-related mental state is used to determine the agent's own suitability to a task. For the purpose of this thesis, a simple hard-coded

data structure was used to represent the self-related mental state. An example of such data structure is discussed in chapter 7, where a particular INDABA implementation is presented in detail.

5.5.2 Task-Related Mental State

Once allocated a task, the agent must store the task information. All information related to a task is stored in the agent's task-related mental state. A task is described by a set of attributes. For the purpose of this thesis, task-related mental state was implemented as a simple data structure, as discussed in chapters 7, 8 and 9.

It is important to note that a simple data structure is not the ideal implementation. A simple hard coded data structure implies prior knowledge about the problem domain. INDABA is not a prescriptive framework, but allows implementations using more flexible mechanisms, such as KQML [66][99] and XML [186].

5.5.3 Society Related Mental State

The implementation of a society related mental state depends on the selected coordination mechanism. For example, in a pure auctioning coordination mechanism, the society mental state consists of only one parameter, namely the cost of an agent. On the other hand, other approaches such as the hierarchical approach (described later in section 6.3.2) and a social networks based approach (refer to section 6.6) require more complex data structures.

For the purpose of the experiments presented in this thesis, the society related mental state was implemented with a social networks based approach in mind. The implementation provides for the creation and maintenance of two distinctive types of social relationships between the agents (described in greater detail in chapter 6), through array-type data structures.

5.5.4 Coordination

Coordination in INDABA is achieved through a cooperative problem-solving process. Neches *et al.* [132], Wooldridge *et al* [198] and Genesereth *et al* [75] divided the cooperative problem-solving process into four main stages:

- **Potential Recognition**: where an agent investigates which agents are capable of executing the task and tasks are allocated.
- **Team Formation**: where agents start to share a common goal and self-organise to form teams to achieve this common goal.
- **Plan Formation**: where an agent or a whole team decides on the division of a goal into sub-tasks and on the allocation of those sub-tasks.
- **Plan Execution**: where an agent or, in the case of a team, agents execute their allocated sub-tasks.

In addition to the four stages above, INDABA introduces an additional stage:

- **Task Success Evaluation**: where awards are distributed to successful team members and penalties are distributed to the unsuccessful team members. These awards are then utilised by a coordination mechanism to increase the affinity of an agent towards the allocated task.

To illustrate the cooperative problem-solving process as implemented in INDABA, an example that uses a social networks based approach for coordination through task allocation is given next. It is important to note that the social networks based approach is purposefully just briefly described as it is presented in greater detail in section 6.6. The emphasis of this example is to illustrate the five stages of the coordination process in INDABA, not the social networks based approach.

This thesis assumes that all members of the team execute the same task. Therefore, in the current implementation of INDABA, the plan formation is omitted as there is no breakdown of a task into sub-tasks. In the illustration of the role of coordination in INDABA, the emphasis is given to the potential recognition, team formation and task

success evaluation steps. The rest of this section overviews these as implemented in INDABA.

5.5.4.1 Potential Recognition

The potential recognition and task allocation functions are implemented as a simple auctioning mechanism. The auctioning mechanism awards a task to the highest bidder, i.e. the agent with the highest score. INDABA does not prescribe the mechanism to calculate the score, but for the purposes of this example it is assumed that the score is calculated using the social networks based approach (refer to chapter 6). An alternative approach, based on CNP [175] was also implemented for the purpose of simulations that are discussed in chapters 7 and 8.

During the potential recognition phase, the known task details are propagated to all agents in INDABA. For the purpose of this illustration, consider task details that consist of three parts:

- ENVIRONMENT_DETAILS, where known environment details are propagated.
- CONSTRAINT, which represents a time constraint as the maximum number of steps that each robot is allowed to execute.
- TASK TYPE, namely scout or forage.

In the initial stages, when social networks are not yet established, INDABA uses a random selection of team members. When social networks are established, INDABA caters for more complex problem domains where there is uncertainty about tasks and the suitability of each agent for a specific task. This is achieved by maintaining social networks that are based on trust and kinship, as described in chapter 6.

By means of developing and maintaining social networks, INDABA provides a mechanism for team selection optimisation, based on historical performance and trust as described in the following sections. Social networks also provide for specialisation

amongst the agents, as opposed to other frameworks that require that all agents' capabilities be known and determined upfront.

To illustrate the propagation of task details, consider that a contracting agent sends task details, as described above, to available robots, with a request to bid. If an agent is busy executing a task or prevented from executing a task (i.e. due to a malfunction), the agent will not respond to the bid. Each of the available robots evaluates its own suitability (affinity) to the task, based on its mental states.

The evaluation of task affinity is a two-step process, based on each robot's history. Firstly, the environment, as given by ENVIRONMENT_DETAILS is identified. Identification is done by encoding the environment according to its known attributes to produce an environment identifier. Secondly, the environment identifier and TASK_TYPE are used to identify the robot's previous experience in the environment identified by the environment identifier, related to the task identified by TASK_TYPE. The experience quantifier, expressed as a ratio between successful task executions and total number of task execution attempts, is then returned as the corresponding robot's bid. After all the available agents have entered a bid, teams are formed as explained in the next section. The potential recognition stage is summarised in algorithm 4.

Potential recognition

If not Auctioning agent

Receive task attributes

Identify environment

Send bid based on history and availability

Else

Send all agents task details

Collect bids

Award task to the highest bidder (team leader)

End

Algorithm 4. Potential recognition

5.5.4.2 Team Formation

The team formation algorithm has two parts, each consisting of a bidding process. The first bidding process is for selection of team leader. The robot with the highest bid is selected and awarded the responsibility of seeing the task to completion. This is either done by the team leader itself, or by a team, selected by the leader. If there are more than one robot with the same highest bid, one robot is randomly selected as the team leader.

The second bidding process is used for selection of additional team members. The additional team members are chosen according to the strength of the social link between potential team member and the team leader. In this example, the strength of social links (based on trust and kinship) in relation to the team leader is used as the bid. The team is formed by selecting the agents with the strongest bids. In a foraging example, if the carrying capacity of the agent with the highest bid is not sufficient, or there are multiple items to be collected within limited time period (as in experiments used throughout this thesis), a foraging team is formed and each member of the team has the same task to collect food (forage). The team formation process is described in p-code in algorithm 5.

Team Formation

Collect bids for team leader

Select team leader as the highest bidding agent

For all agents

    Evaluate strength of the social links between agent and team leader

End For

Select the team members according to the strength of social links

Algorithm 5. INDABA potential recognition and team formation

Once the team is selected, the task is executed. Upon completion (successful or unsuccessful), the performance of the team is evaluated.

5.5.4.3 Task Success Evaluation

The task success evaluation stage is implemented as a function that awards agents that have successfully completed their allocated tasks, with or without the help of other agents. Once task execution is completed, all the successful robots (the robots that have completed the task) are rewarded. The strength of the social links between successful agents that have participated in the team are reinforced by raising the level of trust between the successful team members. For each successful robot in the team, its affinity to the task is improved by means of increasing the ratio between the number of successful task executions and the number of attempts of task execution. The exact reward function is not prescribed, and will be problem dependent.

The next chapter explains how trust is calculated. For the purpose of illustrating the task evaluation process, let trust between two robots R_i and R_j in relation to a task T be defined as *trust* (R_i , R_j , T). The task success evaluation process is then described in algorithm 6.

Task Success Evaluation

For all Robots R_i *in team*
  If Task T successfully executed
    For all remaining Robots R_j *in team that have successfully executed Task T*
      Increase trust (R_i , R_j , T)
    EndFor
  Else
    For all remaining Robots R_j *in team that have not successfully executed Task T*
      Decrease trust (R_i , R_j , T)
    EndFor
  EndIf
EndFor

Algorithm 6. INDABA Task success evaluation

5.6 Summary

INDABA, the new MAS architecture, was presented in this chapter. INDABA consists of four layers, each of which was discussed in detail, with the emphasis on the interaction layer that encapsulates coordination mechanisms in INDABA.

The next chapter discusses the main approaches to coordination. A new approach to coordination using social networks is also presented in the next chapter.

Chapter 6: Coordination Approaches

This chapter provides an overview of the most commonly used coordination mechanisms, classified according to the paradigm of their origin. In section 6.1, definitions of cooperation and coordination are given, as well as clarification of the scope of this thesis with respect to coordination. Coordination approaches inspired by biology are presented in section 6.2, followed by approaches inspired by organisational sciences, described in section 6.3. Basic concepts of social networks are introduced in section 6.4, together with a brief discussion on the applicability of social networks to MASs. An overview of social networks-related research in the field of MASs is given in section 6.5. Lastly, the new coordination approach, based on social networks, is presented in section 6.6.

6.1 Introduction

Agents in a Multi-Agent System (MAS) can exhibit cooperative or competitive behaviours. While competitive behaviour can be encouraged in some computational intelligence approaches, such as evolutionary computing, in robotic applications it is not often desired. In robotics the cost of building a robot is relatively high, thus the evolutionary approach where undesired specimens are discarded, is often not desirable (however, the evolutionary approach can be used in simulations and only the final optimal solution can be implemented physically). It is important to note that cooperative behaviour does not exclude market-based competitive approaches, such as the auctioning coordination technique [175].

The key to a successful MAS is to prevent negative interaction (conflict) and to promote positive interaction (cooperation). In order to promote these two goals, it is necessary to implement a coordination mechanism.

Coordination mechanisms can be broadly divided onto two distinctive groups:

- Emergent - where each agent pursues its own goals, but a coordination-like behaviour emerges through interaction within an environment. This is often the case in swarm robotics [189][177].

- Intentional - where agents actively and intentionally communicate in order to avoid conflict [144].

For the purpose of this thesis, the intentional approach [77] to coordination is followed.

Furthermore, a specific intentional coordination method, task allocation, is the focus of this thesis and emphasis is on task allocation methods as a coordination technique for multi-robot teams.

6.2 Biology-Inspired Approaches – Coordination Perspective

The main advantage of investigating a biology-inspired approach for coordination mechanisms is the existence of coordination in biological systems. In fact, there is an abundance of insects and animals that successfully coordinate, for example ants in an ant colony and wolves in a wolf pack. Biology-inspired coordination mechanisms range from simplistic mechanisms that rely on very limited communication channels (as seen in insect societies) to sophisticated mechanisms that utilise multi-channels of communication (i.e. gestures, sounds and "body language") as observed in mammalian societies.

The diversity of coordination mechanisms requires a clear separation of biology-inspired approaches into two main categories: insect society-inspired and higher mammalian society-inspired. An overview of the main differences between the two approaches follows next.

6.2.1 Overview of Differences Between Insect and Mammalian Societies (Coordination Perspective)

Before considering multi-robot systems inspired by insects (commonly referred to as swarm robotics systems) or mammalian societies, it is useful to consider the

fundamental differences between real, biological agents in insect colonies and those in higher mammalian societies. The differences are discussed from cognitive and social perspectives. A full comparison is outside the scope of this thesis and only the characteristics related to cooperation and coordination are considered:

- ***Ability to learn.*** Insects have a much lower level of self-awareness and their individual learning ability is often non-existent. Insects typically do not develop memories and do not learn from past experience, whereas mammals do learn from past experience.

- ***Communication methods***. Insects have much simpler communication mechanisms that prohibit the exchange of complex messages. Mammal societies usually employ complex means of communication. Another communication-related issue is the localised nature of insect communications. Insects usually communicate by touch and/or chemical reactions [19]. Mammals often use sound. Using sound as a communication method, mammals can communicate over greater distances.

- ***Individualism.*** Most agents are homogenous in insect colonies. In other words, insect colonies are anonymous societies where agents of the same type are indistinguishable. In contrast, kinship and other social relationships are of extreme importance to mammalian societies.

Cooperation is a form of positive interaction between agents. It is a process of working together to achieve a common goal. The requirement for cooperation is the existence of a coordination and/or negotiation mechanism. A view expressed by Matarić [114] is that cooperative behaviours (such as task allocation) based on negotiations require direct communication between the agents.

Direct communication, i.e. when a specific agent is identified and addressed, is not possible in insect societies due to the lack of individualism and insects' limited communication mechanisms. It is important to note that insects do communicate (for

example bees have a dance based communication mechanism), but the communication is limited in range and it is limited in the number of messages that can be communicated. However, it would be wrong to say that insects do not cooperate; they do, but through much simpler mechanisms based on interaction with their environment, using the principle of stigmergy [19]. The stigmergy principle is derived from observations of social insect colonies, such as bees and ants. The process of stigmergy is described as:

"The production of a certain behaviour as a consequence of the effect produced in the local environment by previous behaviour" [19].

Due to the lack of individualism, hierarchies and social relations between agents in insect colonies are virtually non-existent. In mammalian societies, hierarchies and social relations play a fundamental role in the organisation of such a society. Cooperation models are often based on hierarchical and role-based models. Insect colonies, as a cooperation model, were and still are very attractive for applications in robotics [5][98]. The main advantages of swarm robotics are that the insect-like robots are fairly simple to construct; cooperation between such robots should be an emergent property of such a system. Swarm robot teams are usually fault-tolerant (to a degree, because if a sufficient number of agents fails then the whole team might fail).

Coordination in insect-like multi-robot systems was initially done through interaction with the environment, using the principle of stigmergy [19]. The stigmergy-based coordination mechanism is very limited and although emergent cooperative behaviour was observed [168], it has imposed limitations on cooperation methods. The need for a more capable communication mechanism, even in insect-like societies, has been recognised relatively early in research and it has lead to various communication-capable behaviour based multi-robot systems [113]. A summary of the differences between the agent models is given in table 4.

Agent Characteristic	Insect Colonies	Mammal Societies
Communication	Sparse, localised	Complex
Individualism	No	Yes
Learning ability	No	Yes

Table 4. Differences between two biology-inspired agent models

6.3 Organisational Sciences-Based Approach

Since the emergence of more complex work-related structures, it has become a necessity to better organise such structures. This necessity for better organisation gave birth to a range of disciplines under the research field of organisational sciences. Researchers in organisational sciences have concentrated mainly on the two most popular approaches, namely the market-based approach and the hierarchical approach.

Each of these two approaches has its own advantages and disadvantages and each of them has been tried as a coordination technique in the field of MASs. The remainder of this section overviews market-based and hierarchical approaches.

6.3.1 Market-Based Approach

Markets are based on the voluntary exchange of commodities between parties at an agreed price. Market-based coordination is based on the same premise. Markets have many properties but from the point of view of its applicability to robotics, the following properties are of primary interest:

- **Self-organisational property**: Markets are self-organising through a pricing mechanism. This property is highly desirable as it helps with the social approach to MAS design, where agents are viewed as a self-organising society.

- **Demand-supply relationship**: Supply and demand are inseparable and self-regulating. This relationship assumes the existence of two entities: a buyer and a seller.

- **Scalability:** Theoretically there is no limit to the number of participants in a market.

Ideally, when the market functions properly, there is an equilibrium price. The equilibrium price is the fairest cost of the transaction and coordination is nearly optimal. The idea of using market-based coordination, in MAS in particular and in AI in general, has lead to the development of various auction based algorithms. One of the most widely used coordination mechanisms used in AI is the Contract Net Protocol (CNP) [175]. CNP assumes the existence of a buyer, a seller and a price.

With regard to robotics, auction based coordination has mainly been applied to simulated multi-robot teams. An example of such an application is given in [51]. Recently, the first real embodied agent systems that use auction based coordination have appeared [77]. It may be too early to judge the success of such coordination based MASs in real embodied agent applications, but there are concerns with the future of a purely auction based approach.

Firstly, the method for awarding a bid must be determined. It is usually a metric or fitness function that is used to determine a winning bidder. In the case of a task that has not been done before, it is uncertain how to determine a winning bidder.

Secondly, the auctioning mechanism relies on accuracy of the task details that is used by bidding agents to calculate the bid. The information that is submitted to bidding agents is not necessarily accurate or complete.

Thirdly, it is unclear how a purely auction based approach can handle a scenario when a task exceeds the capabilities of each individual bidder. One of the main strengths of MASs is the ability to solve problems that exceed the capability of individual agents.

6.3.2 Hierarchical Approach

Probably the simplest way of coordinating agents is by establishing a relatively strict hierarchical architecture that prescribes the roles for each agent. Such an approach

often assumes a globally coordinated and optimised multi-robot system. The coordination task is done either by a specialised agent [10] or by an agent that has been temporarily assigned the coordination role [103]. The hierarchical approach often uses a symbolic based planner that can provide the optimal solution based on its symbolic environment model.

The hierarchical approach has a number of problems including:

- The inability to create an accurate world model performance [197], and
- it is somewhat contrary to the idea of autonomous agents. Instead of being fully autonomous agents, the agents in a hierarchical approach system are in effect controlled from a central agent.

The hierarchical approach also leads to easier specialisation of agents as agents can have different physical and logical characteristics, adjusted to their specific tasks. Agent specialisation can prohibit optimal load balancing. One of the extreme specialisations is that of MACTA [11][10], where the deliberative (or central planning) coordination agent is a desktop PC, while the team members are physical robots. The side effect of specialisation is that redundancy is reduced. A team member cannot easily replace another team member that has failed because they have different physical and reasoning characteristics.

Specialisation has a positive side: agents can be designed according to the role they are assigned to perform. Specialisation in this context can lead to lowering an agent's complexity and cost as only the required functionality is implemented.

If a centralised coordination mechanism is employed, the role of a reliable communication channel is crucial. In the case of failure of either the global coordinating agent or the communication channel, the hierarchical multi-robot system will fail. The single point of failure characteristic is not desirable. In certain environments (e.g. deep level mining, underwater exploration, electronics emissions saturated battlefield etc.), communication channels can be limited and unreliable. In

such environments, it is improbable that a hierarchical approach would be an effective approach to coordination, as it requires reliable communication channels.

6.4 Social Networks

Traditionally, societies are organised according to socio-economic structures such as markets, hierarchies and networks. Only relatively recently, social networks have been identified as social organisational structures, with one of the first formal definitions of social networks given by Mitchell:

"A (social) network is generally defined as a specific type of relation linking a defined set of persons, objects or events" [125].

Using social networks analysis, social networks can be used to explain why a society, as an entity, functions the way it does. From a social network analysis point of view, a society can be expressed as patterns of relationships between interacting units [195]. Social network analysis can also give insights into emergent patterns, relationships and their implications to a society.

Societies with well-developed patterns of social networks have many advantages. Before exploring these advantages, an introduction to the field of social networks and its terminology is necessary.

6.4.1 History of Social Networks Analysis as a Science

It is outside the scope of this thesis to give a full detailed history of the development of social networks analysis as a science. The reader is referred to [167] for more details. For the purpose of this thesis, only a brief overview is presented next.

Social network theory did not appear suddenly as a unified, complete theory. Instead, social network theory has evolved from the works of various scientists over a period spanning almost a century. Initially, mainly behavioural and organisational scientists were interested in it. At the beginning of the 20th century behavioural scientists have

posed the question "how much do we preconceive objects and concepts and how much do we really perceive them" [167]. One of the answers to this question was the *gestalt* theory by Kohler [97]. Kohler proposed that our perception is defined by organised patterns through which humans interpret the real-world. Kohler's gestalt theory was positively accepted and a number of researchers have expanded on his work, albeit mainly in the field of social psychology.

One of the scientists that has embraced the *gestalt* theory was Moreno [167]. Moreno's research focus was to determine the influence of structures, what he has termed "social configurations", on the psychological well-being of an individual. Examples of such social configurations are concepts of friendship, attraction, repulsion, etc. Moreno's main contribution to the field was the introduction of the concept of "sociometry". Sociometry is a metric function for social relations. Furthermore, he has introduced a "sociogram", which is basically a directed graph representing "social configurations". The improved versions of sociogram are still frequently used to describe social relations. In fact, the sociogram has provided a foundation for graph theory applications to sociological sciences [195]. One way of representing social networks is through such graphs.

Almost at the same time, a group of scientists at Harvard started their work on defining cliques, clusters or blocks within a society in the late 1920s. The most prominent amongst the researchers were Mayo and Werner [167]. Their research focus was different from that of Kohler and Moreno. While, as sociologists, Kohler and Moreno were interested in the application of social networks (or their humble beginnings) to social psychology, Mayo and Werner were interested in applying social networks to anthropology. This diversity of origins of modern social networks analysis emphasises the fact that from the early beginnings, social networks analysis was seen as an interdisciplinary technique.

The Harvard group proposed communities within societies and often informal mechanisms that govern them [167]. By the late 1930s, basically all components of modern social networks analysis were in place, namely relationships, actors, cliques etc. However, a unified social networks theory was still decades away.

In 1969, a Manchester University researcher, J.C. Mitchell, published his work that is widely seen as the foundation of modern social network theory [125]. The majority of concepts introduced by Mitchell are still applicable and in use. Concepts and notations related to social networks, and relevant to this thesis, are defined next.

6.4.2 Social Networks Analysis Concepts

For the purpose of this thesis, only a selection of social networks analysis concepts is presented here. The selected concepts are relevant to the research presented in this thesis and sufficient to support the work done. The selected concepts are used in comments on results presented in chapters 7, 8 and 9.

The selected concepts and definitions are:

- Actors that act semi-independently. Actors are autonomous, yet they are defined and embedded within the society through the existence of social networks. Actors can be seen as nodes in a graph that represents social networks. In MASs in general, and in INDABA in particular, actors are agents. Therefore, the remainder of this thesis uses the term agent instead of the term actor.
- Relationships that link agents to each other. The relationships can either be positively or negatively weighted, and are directed. The social relationships can be seen as indices of a graph that represents a society.
- Social network. A social network is a set of agents and a distinct relationship among the agents.
- Agent society. A society is a representation of a complete set of social networks.
- Cliques or clusters, that are sets of agents defined by existence of strong relationships. A clique is a sub-set of a society, or in graph terms, a sub-graph.

To illustrate the described concepts, consider a society with only one type of relationship, namely the frequency of cooperation between individuals A, B, C, D and

E on a specific project. Table 5 summarises the frequency of cooperation between the agents in the society.

Cooperated	A	B	C	D	E
A	0	27	25	5	0
B	27	0	31	0	3
C	25	31	0	8	7
D	5	0	8	0	35
E	0	3	7	35	0

Table 5. Matrix representing a social network

Based on the relationships given in table 4, figure 13 illustrates a graph that describes the resulting society.

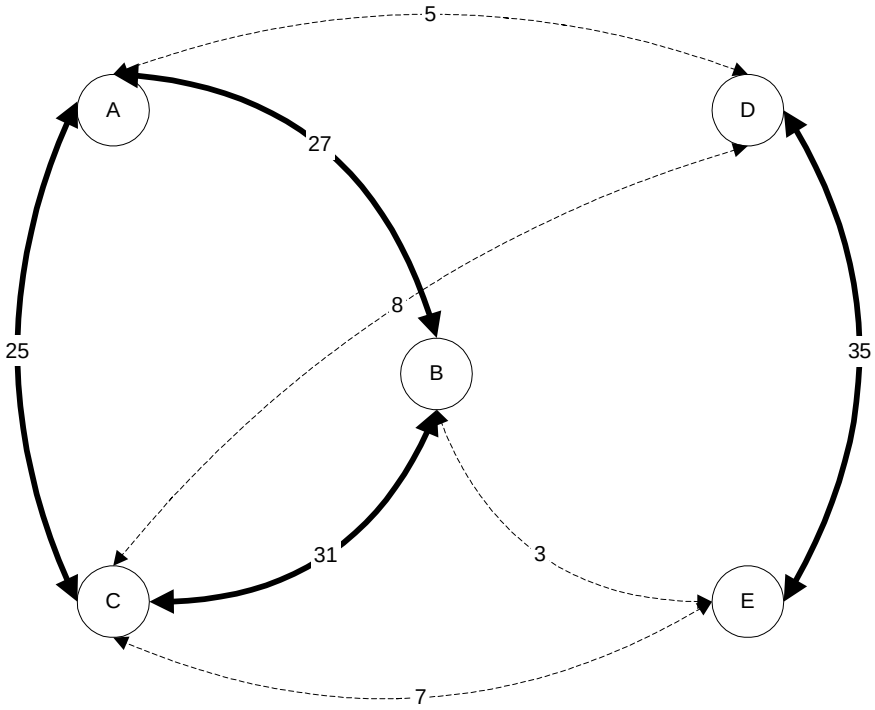

Figure 13. An illustration of a social network representation

Individuals A, B, C, D and E are agents, represented as nodes in the graph. The indices associated with links represent the frequency of cooperation between the agents on the project. The existence of strong relationships between certain members is an indication that those members form a cluster (or a clique). Figure 13 illustrates two clusters (as indicated by bold lines): one consisting of individuals A, B and C and the other consisting of individuals D and E.

Membership to a social cluster or a social relationship is not exclusive: a member can be linked to other members through multiple social relationships. The agents that are linked through a social relationship can be seen as the members of a social group. Examples in human society abound: a person can be a member of a sports club, a university study group and a family. In human societies, social networks are present in everyday interactions but they are not always simple to express and quantify. Key questions applicable to social networks are how the social relationships that define the social networks are formed and how they are maintained [180]. In more complex animal societies, concepts of kinship and trust form a fundamental, but not exclusive, role in the creation of social networks.

It is important to note that this is a simplified representation of social networks in comparison with social networks as observed in the real-world. Real-world social networks are often more complex and have attributes such as direction, durability and intensity [102].

6.4.3 The Importance of Uncertainty in Multi-Robot Teams

Uncertainty about task details is unfortunately one of the realities of implementing any MAS and specifically multi-robot teams in real-world environments. The problem of uncertainty is more evident in robotic applications operating in previously unexplored environments. Those environments are difficult to model, due to the uncertainty about the environment attributes. Furthermore, there is often no previous history of similar applications. Interplanetary robotic exploration is an example of such environments.

The majority of "robots" that were used for interplanetary exploration, starting with the early Soviet Lunokhod series [191] up to the recent NASA's Mars Rovers [187], are not agents or robots in the true sense of the definition of an agent (refer to section 2.2.3). These vehicles are not autonomous, but tele-operated from Earth, leaving just a basic interaction with its environment to their internal mechanisms.

While "robots" within the inner Solar system can be tele-operated due to the fact that the delay caused by the finite speed of radio signal propagation is within (barely) acceptable limits, the same method of exploration will not be possible for the further reaches of the Solar system and beyond. Interplanetary exploration is just one of the problem domains that will benefit from the evolution of more autonomous, self-organising robotic systems.

On other hand, biological systems, such as teams of animals or humans, generally cope with uncertainty. In other words, teams of animals or humans, when put in different environments and faced with unfamiliar tasks, generally achieve their goals. The view proposed in this thesis is that one of the contributing factors is the existence of social networks that define a team and the structures within it.

6.4.4 The Applicability of Social Networks to Multi-Robot Teams

A major advantage of societies with multiple, well-established social network is that they are flexible enough to allow the best team for the task to be selected by using the most appropriate social relationship that in turn defines the social group. For example, if a task involves participation in some sport, the member belonging to the sports club that practices that sport should be used to form a team. Affinities between social groups can also play a significant role. If there is uncertainty about the task and no social group satisfies the demands of that task, then the group with the highest affinity for the task should be selected. The existence of such affinity relationships between social groups is very important. Affinity between social groups is especially important when there is uncertainty (lack of detailed information) about the task or in the case where the best candidates for a task are already allocated to another task. In the case that the best candidates are not available, the "next best" candidates must be selected. The "next best" candidates are the members of the social group with affinity to the optimal social group.

It has been noted in section 6.4.2, that social networks are sets of agents and relationships between them. Agents are members of a society and robots in multi-

robot teams can be viewed as members of a society. Zini, for example, defines a MAS as a society of agents [204].

With the previous definition in mind, a multi robot-system can be defined as:

"…a social and cognitive entity with a relatively identifiable boundary, that functions on a relatively continuous basis through the coordination of loosely interdependent, cognitive and autonomous agents" [22].

By considering a multi-robot team as a society, social networks between robots can be identified and analysed. The knowledge obtained from the social networks analysis can then be used to describe multi-robot teams, as well as to predict and coordinate the behaviour of the team as a single entity.

Social relationships within a society are often very complex and can be multidimensional. More often than not, there can be more than one social relationship between two agents. In human societies, it is easy to grasp the wealth of relationships with all members of societies being linked to others.

For the purpose of this thesis, the approach taken is to initially consider higher mammalian societies and to isolate only a few applicable relationships. This is by no means an exhaustive approach, but rather an exploratory approach. The social network-based approach, as presented in this thesis, is by no means limited to the number of relationships that each agent can have. However, the implementation of the social networks based approach developed for the purpose of the simulations presented in this thesis is limited to two social relationships only, namely kinship and trust.

6.5 Related Work

Higher mammalian societies, and specifically human societies, have inspired research related to the applicability of the coordination mechanisms in MASs. During the 1990s, a number of researchers were involved with various society-inspired MAS-related research.

Social networks are an integral part of such societies and although there is no directly related work done on utilising social networks for task allocation as presented in this thesis, it is important to overview existing research efforts.

The social network-related research efforts can be broadly divided into three main categories:

- Research interested in social hierarchies as coordinating mechanisms.
- Research in modelling higher mammalian societies in order to better understand the subtle relationships that exist in them, with a view to be possibly used as coordination mechanisms.
- Research into the use of social networks for trust propagation in MASs.

6.5.1 Social Hierarchies and MAS Applications

Social hierarchies have been of interest to researchers from the early days of DAI research. The early work related to decentralised AI with application to multi-robot teams can be traced back to the work of Luc Steels in 1990 [177], although Steels was mainly interested in societies which were less complex than mammalian societies (insect societies).

The higher mammalian societies, such as packs of wolves and troops of chimpanzees, have also been investigated [149]. The hierarchies within these societies have inspired research at MIT [69] to explore the benefits of hierarchies in MASs for the purpose of tasks such as streamlining inter-agent negotiations and forming alliances between agents.

6.5.2 Modelling Societies

Using agents to model societies is becoming increasingly popular. Agent systems have been used to model the societies of primates [149], using tools such as MACACA [100], and even early human societies [55].

Modelling of social relationships between agents in a MAS, specifically between robots in multi-robot teams has been the focus of research headed by Dautenhahn [47][48]. A survey of socially interactive robots can be found in [69].

6.5.3 Social Networks for Trust Propagation In MAS

The concept of trust is very important in interaction between agents in a MAS. If coordination and/or cooperation is required, an agent makes a decision based on its perception of other agents' capabilities. The ideal situation is that each agent in a MAS is fully aware of all other agents' capabilities and their current status. More often than not this is not possible and in these situations an agent must trust the other agents' estimates of their own capabilities. For a detailed survey of trust-related research in the field of MASs the reader is referred to [163].

One of the methods for establishing trust in an agent's capabilities is through trust propagation via the process of querying trusted agents about the capabilities of the agent whose credentials need to be established. Social networks provide a mechanism for trust propagation. More on utilisation of social networks in trust propagation can be found in the work of Yu *et al* [201] and Schillo *et al* [164]. It is important to note that trust as defined in work of Yu is related to information systems security issues, while in this thesis the trust is related to agent capabilities.

6.6 Social Networks Based Approach

The new social networks based approach to coordination presented in this thesis uses task allocation as a coordination mechanism. The basic concepts and origins of the social networks based approach for coordination are discussed next.

6.6.1 The Biology Origin

The social networks based approach has its foundations in the observed similarities between higher mammalian societies and multi-robot systems. In both biological (e.g.

a wolf pack) and artificial (multi-robot team) systems, there is often the need for cooperation. In this section, a conceptual comparison is given between a multi-robot team and a pack of wolves.

Wolves are social animals that are organised into packs, governed by strong male and female animals (alpha male and alpha female) [133]. A wolf pack is characterised by the existence of a strong social hierarchy [133]. A wolf pack is usually a family unit, reflecting the existence of a strong kinship relationship between the pack members. A wolf pack is a very effective hunting team and can bring down prey much bigger than an individual wolf could.

The comparison between a wolf pack and INDABA will be made in relation to the five steps of the cooperative problem-solving approach as proposed in INDABA (refer to chapter 5), namely potential recognition, team formation, plan formation, plan execution, task evaluation and recognition.

In terms of the adopted robotic MAS taxonomy (refer to section 4.1), a wolf pack and INDABA can be compared as in table 6 (note that the characteristics of INDABA are described as the upper limit, not as a particular implementation, as INDABA is only a framework):

Team Characteristic	Wolf Pack	INDABA
Size of Team	LIM	INF
Communication Range	NEAR	INF
Communication Topology	BROAD	GRAPH
Communication Bandwidth	MOTION	INF
Collective Reconfigurability	DYN	DYN
Processing Power of a Team Member	TME	TME
Collective Composition	HET	HET

Table 6. Comparison of a wolf-pack and INDABA

6.6.1.1 Potential Recognition

When successful task completion requirements exceed the capabilities of an agent, a need for cooperation is recognised by members of the society. In the case of a wolf pack, wolves often hunt prey that is too big to be hunted by a single member. The need for cooperation is recognised and a wolf pack hunts as a team.

In the case of multi-robot teams, tasks that exceed the capabilities of a single robot abound. Box pushing [199][107] is one of the well-known problems, as well as foraging under a time constraint, where a single robot cannot complete the task in a prescribed time period, while a multi-robot team can.

For both biological (wolf pack) and artificial systems (multi-robot team), the need for cooperation is recognised. The potential recognition leads to the next step, i.e. team formation.

6.6.1.2 Team Formation

In a society a team is formed according to the relationships between its members. Considering a wolf pack, the distribution of labour is according to the hierarchical structure of the pack. The hunt is lead by the alpha male and alpha female, as they are the most efficient hunters.

If neither the alpha male nor the alpha female is capable of leading the hunt, the next most capable members will lead the hunt (a beta male or beta female will assume their position).

Considering heterogeneous multi-robot teams, similarities with a wolf pack are many. The most capable members of a multi-robot team are selected for a task. When the most suitable robots are not an acceptable choice (they might be cost-prohibitive or unavailable or malfunctioning etc.), the next most suitable robot will be selected, according to a social network relationship (e.g. “next of kin” or another member of a social group).

6.6.1.3 Plan Formation and Plan Execution

The plan formation step is not always applicable to packs of animals and multi-robot teams, and is largely ignored for the purpose of this comparison. It is sufficient to note that members of a society can have a specialised role that they perform and that a plan formation should take such specialisation into account.

The same applies to the plan execution step of a cooperative problem-solving process. Plan execution is not relevant to this comparison of similarities between biological and artificial societies as the focus is on task allocation.

6.6.1.4 Task Evaluation and Recognition

It might not be obvious, but a form of reward and punishment mechanism can be found in both a wolf pack and multi-robot teams (if it is so designed and programmed).

Considering a wolf pack, if the hunt was successful, the social hierarchy will be updated by strengthening the existing relationships. Furthermore, because the feeding order is dictated by the social hierarchy, the alpha male and female will eat first and eat the best parts of the hunted animal, in turn maintaining their physical supremacy over the rest of the pack.

In a multi-robot team (if so programmed) the agents that succeed in task execution will be rewarded and their affinity to the task will be increased. The team leader will strengthen its affinity to the task and maintain its team leader status.

However, if the hunt fails, probably nothing will happen immediately for the wolf pack. However, if the alpha male fails to feed the pack for extended periods of time, its social position can deteriorate to the extent that it is challenged by a beta male.

The same principle is applied to multi-robot teams. If a team leader repeatedly fails, its affinity to a task is decreased. It can happen that at a certain point in time the team leader is no longer the top-scoring agent in the team and it stops being the team

leader. In other words, it may happen that over time, the team leader's bid to secure a task may be insufficient, in which case a different robot may win the bid and select a new team.

6.6.2 Comparison to Other Task Allocation Coordination Mechanisms

Approaches other than INDABA have been developed to use task allocation as a coordination mechanism, for example MURDOCH [77] and BLE [196].

BLE is based on the subsumption architecture and uses a port-inhibiting strategy for task allocation, where a robot can decide that it is the best eligible for a task. If a robot is eligible for the task, the robot can inhibit a communication port, effectively seizing control. MURDOCH, a market-based approach, uses a more traditional approach, an auctioning mechanism that governs task allocation, again based on a robot's own estimate of its capabilities. An extensive review of multi-robot task allocation mechanism can be found in [76].

While each robot is an autonomous agent in MURDOCH and BLE, the agents are basically unaware of other members of the society. The social networks approach presented in this thesis is different, because it relies on agents to belong to social groups and that agents maintain social links among themselves.

Task allocation in INDABA consists of selecting a team leader. This can be done by either using an auctioning mechanism or agent's historical performance on same or similar tasks. Once a leader is selected, a team is formed based on the strength of agents' relationships to the team leader and the task. A team is formed using a scoring system which takes into consideration all applicable relationships to the task in question. Based on relationships and task affinity each agent is given a score. The society members with the highest scores form the team together with a team leader.

6.6.3 Definitions and Notification

To describe the social networks based approach to task allocation in more formal terms, the following definitions are necessary.

Let T_k be a task that needs to be allocated, where $k = 1,..,K$ with K the number of tasks. If there are n known attributes of task T_k, then task T_k can be represented by an n-tuple, $(T_{k1}, T_{k2},..., T_{kn})$, where $T_{k1}, T_{k2},..., T_{an}$ are the n attributes that define task T_k.

The value of attributes $T_{k1}, T_{k2},..., T_{kn}$ can either be binary, discrete or continuous valued.

Let A_x be an agent, whose suitability to task T_k needs to be evaluated, where $x = 1,..,X$ with X the number of agents in the society. If there are m known attributes of agent A_x, then agent A_x can be represented by an m-tuple $(A_{x1}, A_{x2},..., A_{xm})$ where $A_{x1}, A_{x2},..., A_{xm}$ are the m attributes that define agent A_X.

The value of attributes $A_{x1}, A_{x2},...,A_{xm}$ can either be binary, discrete or continuous valued.

Let A_{lk} be the agent whose applicability to task T_k is the highest. Then agent A_{lk} is a team leader. Each leader has m known attributes, $A_{lk1}, A_{lk2,...}, A_{lkm}$. The leader is represented by a m-tuple $(A_{lk1}, A_{lk2},..., A_{lkm})$ where $A_{lk1}, A_{lk2,...}, A_{lkm}$ are the m attributes that define agent A_{lk}.

Let there be I relationships between agents in the society. Then relationships between agents A_{lk} and A_x in relation to task T_k are denoted as $R_i(A_{lk}, A_x, T_k)$ for $i = 1,.., I$. R_i is a function normalised to the interval [0,1], that is, $R_i:(A_{lk}, A_x, T_k) \rightarrow [0,1]$.

It is important to note that not all functions that model relationships require T_k as an input. For example, kinship is independent of a task under consideration, and therefore the kinship relationship depends only on the two agents involved.

With these definitions in mind, it is possible to define a fitness[2] (or scoring) function for an agent A_X and the team leader A_{lk} in relation to given task T_k as

$$F_{xk}(A_{lk}, A_x, T_k) = \Sigma_{i=1...I}(1-k_i) R_i(A_{lk}, A_x, T_k). \quad (6.1)$$

where $\Sigma_{i=1...I} k_i = 1$. It is interesting to note that if A_{lk} is omitted from equation (6.1), then the remaining function F_{xk}', given as

$$F_{xk}'(A_x) = \Sigma_{i=1,..,I}(1-k_i) R_i(A_x, T_k) \quad (6.2)$$

is used by agent A_x to estimate its own eligibility to task T_k.

6.6.4 The Social Network Task Allocation Algorithm

The social networks task allocation algorithm is outlined below in general terms. A specific implementation that uses two social relationships is presented in greater detail in section 7.1. The social networks task allocation algorithm can be seen as an enhanced or augmented auction based task allocation algorithm. The bid is a function of the strength of social networks. The algorithm itself is surprisingly simple, and the key to its efficiency is in keeping the relationships up to date. The relationships can be stored either in a central repository or they can be distributed, with each agent maintaining its own social networks.

The advantage of a central repository system is that it is simpler to implement, but on the other hand it does require reliable communication channels between all the agents in the society and the central repository.

The distributed model is more applicable to robot teams and, in general, closer to the true notion of an agent. It does, however, require more complex implementation than a central repository system.

[2] Please note that in the context of this thesis, the notion fitness function is different from the concept of fitness function as used in evolutionary computing, i.e. it does not influence the survival of the agents in the society.

In the new proposed architecture, INDABA (refer to chapter 5), each agent maintains its own relationships data. The relationships data is created, stored and maintained in the interaction layer. From the robotic application point of view, the advantage of this approach is that an agent can rely on its internal social network to find the best candidates in its local environment, even if there is no complete and reliable communications with the rest of the agents.

The algorithm consists of four main steps:

- task detail propagation component
- the selection of a team leader most suitable to the task
- selection of the remaining team members
- task evaluation and social network maintenance

Each of these steps is described next.

6.6.4.1 Task Details Propagation Component

Once the task details are known, they are propagated to all available agents in the society. An external party, such as a user of the system, can either give the details of the task, or the task details can be obtained through the agent's exploration of its environment.

An example of the latter approach is a heterogeneous robot team where a specific robot performs the role of a scout and collects the information about the environment. The scout collects all the details about the environment using its own sensor suite and sends the details to the rest of the team for task allocation.

The task details T_k, represented as the n-tuple, $(T_{k1}, T_{k2}, \ldots, T_{kn})$, needs to be propagated to all participating members of the team. The propagation of task details may utilise any available communication protocol or method. The implementation of task propagation is not prescribed by INDABA and can take any form, from a simple

binary coded string of predetermined length (where values correspond to the sensor readings of a scout) to much more flexible approaches such as KQML [66][99] or XML[186].

For truly unknown environments, where even metadata about the environment is not available, KQML together with a semantic descriptive language, for example Knowledge Interchange Format (KIF) [75], would be an advised approach. To illustrate, consider a scout in an unknown environment. If the scout discovers new concepts, those newly discovered concepts (metadata) can be described using KIF and propagated to the rest of the agents in the society using KQML.

Propagation of task details can be done either by a centralised entity (such as an external supervisory program, an approach similar to a supervisor in MACTA) or an agent can initiate the propagation of task details, in which case the agent becomes a managing agent. While the agent-initiated approach is advisable, it is more complex to implement. The implementation of propagation of task details is made according to the environment requirements and agent capabilities.

6.6.4.2 Team Leader Selection

Team leader selection can again proceed in at least two ways: either an agent can submit its own task affinity evaluation or an agent can be evaluated by an external supervisory entity. Considering leader selection, once the task details T_k are received, scoring takes place based on agent attributes and scoring function F_{xk}' (refer to equation (6.2)). The agent with the highest F_{xk}' is selected as the team leader.

Team leader selection is not social network related in its true sense (links with the other members of the society are not examined or utilised in selection), but it can rely on either direct matching of attributes to the task details (if possible) or to historical data (if available). If direct matching is not possible nor historical data available (which is the case when the task is executed for the first time), then an alternative selection method must be used, using a different scoring function.

As an example of an alternative team leader selection method, the leader can be selected randomly. The main advantage of a random selection method is that all agents are given an equal chance for task selection. However, using a random selection method, there is no guarantee that the selected team leader is capable of executing task of team leader. Alternatively, the cheapest member, determined using a cost function (if implemented), can be selected.

Both the social networks team leader selection method and an alternative team leader selection method can be combined in a single algorithm. If there is no historical data, an alternative selection method that uses a different scoring function $F_{xk}^{''}(A_x)$ can be utilised, otherwise social networks selection based on scoring function $F_{xk}^{'}(A_x)$ is used, as illustrated in algorithm 7.

$A_{lk} = A_1$
If historical data available or direct matching possible
  $F_{xk}(A_X) = F_{xk}^{'}(A_X)$
Else
  $F_{xk}(A_X) = F_{xk}^{''}(A_X)$
End If
For all agents A_x *in society S*
    If $F_{xk}(A_x) > F_{xk}(A_{lk})$
      $A_{lk} = A_x$
EndFor

Algorithm 7. Team leader selection in social networks based approach

6.6.4.3 Team Selection

Once a team leader has been selected, the rest of the team is selected. At this stage social networks play a crucial role.

The team member candidates are not only evaluated in relation to the task, but also based on their relationships to the team leader. This may look counter-intuitive, but it is not enough that a team member has an affinity to the task, the candidate must also be capable of working together with the team leader.

The relationships to the team leader are the crucial part of the algorithm and form the premise of the whole social networks-inspired approach. Agents are not individual, independent entities, but are defined in relation to the other members of the society. For team selection, team member candidates are also evaluated according to their ability to cooperate with the team leader, based on previous history (trust) and similarity to the team leader (kinship). In analogy to human societies, an agent is evaluated on how good it is as a "team player".

Team selection can be done by a team leader either according to its existing relationships to the other team members or the relationships can be recalculated prior to team selection. For both cases the algorithm is basically the same as summarised in algorithm 8.

While team T less than TeamSize
  $A_N = A_1$
  For all agents A_x *in society S*
    If A_x *not allocated to team T and* $F_{xk}(A_{lk}, A_x, T_k) > F_{xk}(A_{lk}, A_N, T_k)$
      $A_N = A_x$
  EndFor
  Add A_N *to team T*
EndWhile

Algorithm 8. Team selection in social networks based approach

6.6.4.4 Social Networks Maintenance

Once task execution finishes (successfully or not), the social networks need to be updated. Each member of the team needs to be evaluated and its relationships updated.

The exact method of updating the relationships is not prescribed by INDABA and can take the form of simply increasing a counter of successful or unsuccessful executions related to a particular task, or towards a particular team member. More complex methods can also be implemented.

Assuming that methods of updating agent relationships are given as *strengthen* and *weaken*, the social networks maintenance algorithm is summarised in algorithm 9.

For all agents A_x *in team* T
  For all $R_i(A_{lk}, A_x, T_k)$
    If T_k *completed*
      strengthen $R_i(A_{lk}, A_x, T_k)$
    ElseIf
      weaken $R_i(A_{lk}, A_x, T_k)$
    EndIf
  EndFor
EndFor

Algorithm 9. Social network maintenance

6.7 Summary

This chapter started with an overview of biologically inspired approaches to coordination. This discussion was followed by an overview of the two main approaches to coordination in MASs, which are based on organisational sciences, namely the market-based and hierarchical approaches.

The remainder of the chapter introduced the concept of social networks and presented a new approach to coordination in MASs. The new approach was based on social networks, and a modification for application in multi-robot teams was presented.

The next chapter presents the implementation of the social networks based approach within the INDABA framework, applied to simulated robots in an abstract simulated environment.

Chapter 7: Experiments in an Abstract Simulated Environment

The focus of this thesis is on an architecture for robotic systems and on the use of social networks as a coordination mechanism. The development of a realistic multi-robot simulated environment is not a trivial task and a decision was made to verify the validity of the INDABA architecture and coordination mechanism through a simpler, abstract multi-robot simulator. The abstract simulator set-up and its scope limitations are presented in section 7.1. The abstract simulator has two main algorithmic components. The first of the two is the task allocation and team formation algorithm, presented in section 7.2. The second algorithmic component is the task execution and task evaluation algorithm, presented in section 7.3. The simulations of uncertainties about task details are presented in section 7.4. The majority of simulations that were conducted for the purpose of this thesis are presented, together with the results and a discussion of these results in section 7.5. A summary of the results, as presented in section 7.6, concludes this chapter.

7.1 Scope Limitation and Simulation Set-up

While previous investigations focussed on social networks and coordination in abstract terms [154][156], this chapter the focuses on an abstract simulator. The main purpose of the abstract simulator used in this thesis is to provide a platform for simulations to explore the proposed social networks based approach to task allocation. The simulated environment implements only the two upper layers of the INDABA framework (see chapter 5), namely the interaction and the deliberator layers. The other two layers, sequencer and controller, are grossly simplified in this simulation. Such approach is justified since only the upper two layers of INDABA are relevant to coordination in general and to the task allocation problem in particular. In the experiments presented in this chapter, a population of fifty agents was randomly created. All agents are defined by the same set of attributes, with the attribute values

randomly selected from the domain of each attribute. The agent attributes and possible attribute values are given in table 7.

AGENT ATTRIBUTE	POSSIBLE VALUES
LOAD	LOAD_SMALL
	LOAD_NORMAL
FALSE FOOD SENSOR	PRESENT
	NOT_PRESENT
DRIVE	DRIVE_WHEEL
	DRIVE_TRACK
	DRIVE_LEG
SPEED	SPEED_LOW
	SPEED_MEDIUM
	SPEED_FAST
DETECTION RANGE	DETECTION_NORMAL
	DETECTION_LIGHT_ONLY
	DETECTION_ADVANCED
POWER	POWER_TETHERED
	POWER_SOLAR
	POWER_BATTERY

Table 7. Simulated agent attributes and possible attribute values

Environments are defined in a similar manner to agents. Each environment is defined by the same set of attributes that represents physical characteristics of the environment. Environments were also created randomly, by assigning random values to the environment attributes. The environment attributes and valid attribute values are given in table 8.

ENIVRONMENT ATTRIBUTE	POSSIBLE VALUES
TERRAIN	TERRAIN_NORMAL
	TERRAIN_ROUGH_AREA
LIGHT	NO_SHADED_AREAS
	SHADED_AREAS
FOOD_DISTANCE	FOOD_FAR
	FOOD_CLOSE
FOOD_TYPE	FOOD_LIGHT
	FOOD_HEAVY
	FOOD_MIXED

Table 8. Simulated environment attributes and possible attribute values

The following rules define interactions that may occur between the environment and agents:

- If the robot LOAD attribute is LOAD_SMALL, it cannot load food that has FOOD_WEIGHT attribute FOOD_HEAVY.

- If the robot has as a POWER attribute value of POWER_SOLAR, it cannot move in an environment area that is in the shade.
- If the robot has POWER attribute POWER_TETHERED, it is limited in range.
- The detection range is reduced in the shaded area and if the robot DETECTION_RANGE attribute has value DETECTION_LIGHT_ONLY, the robot cannot detect objects at all.
- The number of steps required for robot movement is a function of the environment terrain (if it is in TERRAIN_ROUGH_AREA), drive and speed. For example, if a robot is in rough terrain area, and if the robot's drive attribute is track or wheel, then the number of required steps is increased by 30% and 60%, respectively. The values of 30% and 60% are arbitrarily chosen. In future work, these values will be changed according to observed decrease in performance of a real robot.
- If a robot is not equipped with FALSE_FOOD_SENSOR and the environment FOOD_TYPE is FOOD_MIXED, it has a 30% chance of picking up false food and failing the task.

For the purpose of simulations in the abstract simulated environment, two types of social relationships were implemented. The implemented social relationships were based on the concepts of trust and kinship, which are the fundamental concepts in the formation of social networks in complex animal societies. These concepts are defined next.

7.1.1 Kinship

Kinship is defined as the similarity between the simulated robots. A simple metric that quantifies kinship, $d(R_1, R_2)$, between robots R_1 and R_2 is calculated as

$$d(R_1, R_2) = \Sigma_{i=1,..,N} d_i (A_{1i}, A_{2i}) / N \qquad (7.1)$$

where N is the number of robot attributes, $d_i (A_{1i}, A_{2i})$ is a normalised metric function in the range [0, 1] between the i-th attribute of robots R_1 and R_2, and A_{1i} and A_{2i} are

the values of the *i*-th attribute of robots R_1 and R_2 respectively. Function $d_i\ (A_{1i}, A_{2i})$ quantifies the difference between the attributes A_{1i} and A_{2i}.

7.1.2 Trust

Let task T be defined by attributes $T_1, \ldots, T_m$, where m is the total number of attributes that define task T. Trust is then defined as a 3-tuple, $t\ (R_1, R_2, T)$, where R_1 and R_2 are robots, and T is a particular task. Tuple (R_1, R_2, T) quantifies the reliability of robot R_1 in relation to R_2, based on the historical performance related to task T that has involved both robots; in other words, how much trust R_1 has in R_2 in helping to complete task T. Trust is calculated as a ratio between the number of successful task executions and the number of task execution attempts. Note that in the approach presented in this thesis, trust is a symmetric function, i.e. $t\ (R_1, R_2, T) = t\ (R_2, R_1, T)$. The value is normalised to the range [0, 1]. Note that $t\ (R, R, T)$ represents the historical performance of robot R in relation to task T; in other words, the trust that robot R has in its own abilities.

For the purpose of simulations presented in this chapter, the strength of a social link, $s\ (R_1, R_2, T)$, between two agents is defined as :

$$s\ (R_1, R_2, T) = kd\ (R_1, R_2) + (1-k)t(R_1, R_2, T) \qquad (7.2)$$

where $k \in [0,1]$.

7.2 Task Allocation and Team Formation Algorithm

The algorithm starts with task details propagation. During this phase, the known task details are propagated to all agents. For the purpose of this thesis, task details for the simulations presented in this chapter consist of three parts:

- ENVIRONMENT_DETAILS, where known environment details are propagated. The environment details are implemented as a set of attributes. The attributes have discrete values, as given in table 6.
- TASK TYPE, namely scout or forage.

- CONSTRAINT, which represents a time constraint, as the maximum number of steps that each robot is allowed to execute.

The first task to be executed is that of scouting. To illustrate the propagation of task details, consider that a contracting agent sends task details, as described above, to all available robots with a request to the bid for a scouting task. If an agent is not available, the agent will not respond to the bid. Each of the available robots evaluates its own suitability (affinity) to the task, based on task attributes and previous experience (as explained below). This part of algorithm is based on general team leader selection algorithm 7.

The evaluation of task affinity is a two-step process, based on each robot's history. Firstly, the environment, as given by ENVIRONMENT_DETAILS is identified. Identification is done by means of encoding the environment according to its known attributes to produce an environment identifier. Secondly, the environment identifier and TASK_TYPE are used as indices in a table that stores the robot's previous experience in the environment identified by the environment identifier, related to the task identified by TASK_TYPE. The experience quantifier, expressed as a ratio between successful task executions and total number of task execution attempts, is then returned as the robot's bid. After all available agents have entered a bid, the scouting task is allocated to the agent with the highest bid.

The role of a scout is to explore the environment and to determine the values of the environment attributes as given in table 6. It is important to note that the scout might not be able to get accurate information about the environment. For example, if a path from the start point to the food concentration area does not cross a "shaded" area, the scout will not set the LIGHT environment attribute to SHADED_AREAS.

Once the scout completes its task, task details based on the observed environment are then propagated to all agents and the algorithm repeats the bidding process as described above and in more general terms, as given in section 6.6.4.2. The robot with the highest bid is selected as the team leader and given the responsibility of seeing the task to completion. If a number of robots have the same bid value, one is randomly

chosen. If the task exceeds the capability of the team leader, then the team leader selects a team that can execute the task. The team selection algorithm is based on algorithm 8.

The team leader forms a team according to the social relationships between the team leader and members of the team. Team selection starts by calculating all social relationships (trust and kinship). Trust is obtained by using a trust function, as described in section 7.1. Kinship can be obtained either by requesting a full set of attributes from a potential team member and calculating the kinship value, as given in equation 7.1, or by using the values that were predetermined and stored in each robot's kinship table. For the purpose of this thesis, predetermined kinship values were used. Once trust and kinship are calculated, the team is then formed by selecting those agents with strongest social relationships to the team leader. The task allocation and team formation process is described in algorithm 10.

Potential recognition

If not Auctioning agent

Receive task attributes

Identify environment

Send bid based on history and availability

Else

Send all agents task details

Collect bids

Award task to the highest bidder (team leader)

End

Team Formation

If team leader

For all agents

Evaluate strength of social link between agent and team leader

End For

Select the team members according to the strength of social links

EndIf

Algorithm 10. Potential recognition and team formation processes

The foraging task is a loosely coupled task, in the sense that each robot can execute its own task without relying on other robots, and each member of the team executes the same task in parallel.

Once the team is selected, the task is executed. Upon completion (successful or unsuccessful), the success of the task is evaluated. The implementation of the task execution and evaluation algorithms is discussed next.

7.3 Task Execution and Task Evaluation Algorithm

Task execution is simulated in this chapter. For the purpose of this simulation, tasks were simulated and evaluated according to the rules outlined in section 7.1. The simulated task execution is discussed next.

7.3.1 Task Execution

In order to simulate interaction with the environment, each robot is provided with a full set of environment attributes (not the potentially inaccurate set of environment attribute values as observed by the scout) and simulated execution takes place using the full environment set of attributes. Each robot action requires a number of steps. The number of steps that it will take the agent to execute a task is calculated, based on the robot's attributes and environment and interaction rules (as described in section 7.1.). For example, the number of steps for a robot to move from coordinate (x,y) to coordinate (x+1, y) is dependent on the robot's attributes SPEED and DRIVE, and the environment at coordinate (x+1,y). An execution cycle is the completed simulated execution of an allocated task, after which the success of the task is determined. The number of allowed steps limits the lifetime of an execution cycle. If a robot has not yet completed the task when the number of allowed steps is exceeded the robot is considered to have failed in its task.

7.3.2 Task Evaluation

As mentioned, if a robot exceeds a predetermined threshold, then the robot fails in its allocated task. The threshold is selected as the average number of steps required for successful task executions, averaged over a randomly chosen observation period (10 simulations, each consisting of 100 execution cycles) and a team of five robots, randomly chosen from the population of fifty robots.

If the execution of a task was successful, all the successful robots are rewarded. The strength of the social links between the successful agents that have participated in the same team will be reinforced, by raising the level of trust between the successful team members. For each successful robot R_j in the team, its trust rating $t\ (R_j\ ,\ R_k\ ,\ T)$ are improved by means of increasing its ratio between number of successful task executions and number of attempts of task execution. The increase is an additive increase. In other words, the number of successful task executions is increased by one. The trust rating $t\ (R_j\ ,\ R_k\ ,\ T)$ is related to task T and other successful members of the team R_k , where $k = 1,..,\ n$, and n is the number of successful team members. Task success evaluation is a function that awards agents that have successfully completed their allocated tasks, with or without the help of other agents. Each agent R_j also keeps its own trust $t\ (R_j\ ,\ R_j\ ,\ T)$ which represents its own historical performance, relative to a particular task's details. This in turn tracks an agent's affinity to a particular task type.

7.4 Simulating Task Details Uncertainty

The view adopted in this thesis is that uncertainty is unavoidable in real-world robotic applications. All experiments done for the purpose of this thesis include varying degrees of uncertainty. Basically, there are two causes of uncertainty that affecti the experiments, namely uncertainty due to environment variations and uncertainty due to the initial robot positioning. These causes of uncertainty are described next.

7.4.1 Uncertainty due to Environment Variations

As described in section 7.1, each environment is defined by a set of attributes. It is important to note that even if two environments have the same attribute values, as defined in ENVIRONMENT_DETAILS (section 7.1), the two environments are not necessarily the same. Even though environment attribute values may be the same, the location of rough terrain, shaded areas and food may differ due to random creation of these aspects. Random creation of environments brings a level of task uncertainty. Uncertainty due to environment variations is the main contributor to uncertainty in the experiments that are presented in section 7.5.

7.4.2 Uncertainty due to Initial Robot Positioning

Uncertainty due to initial robot positioning has a lesser influence on task execution than uncertainty due to environment variations. Uncertainty due to the initial robot positioning is caused by randomly initialising each robot's position on different coordinates of the home area (which is defined as the lower right corner of the environment). Random initial positions have an influence on the number of steps required to reach the food area and also contributes to uncertainty about task execution.

To illustrate the influence of uncertainty due to initial robot positioning to successful task completion, two simulations were done. The first simulation was done with a constant initial position for all robots, while the second simulation was done with variable initial robot positions but within the limits of the home area. Each simulation consisted of 100 execution cycles. The team size was limited to six team members.

The results are presented in figure 14. In the case of a constant initial position for robots, optimal team selection was achieved early in the simulation - in the first 20 execution cycles. In the case of variable initial robot positions, team selection was achieved later in the simulation. The results were never as good as in the case of the constant initial position. However, in case of variable initial robot positions, once a

team was selected, the selected team was more resilient to change in the initial positioning.

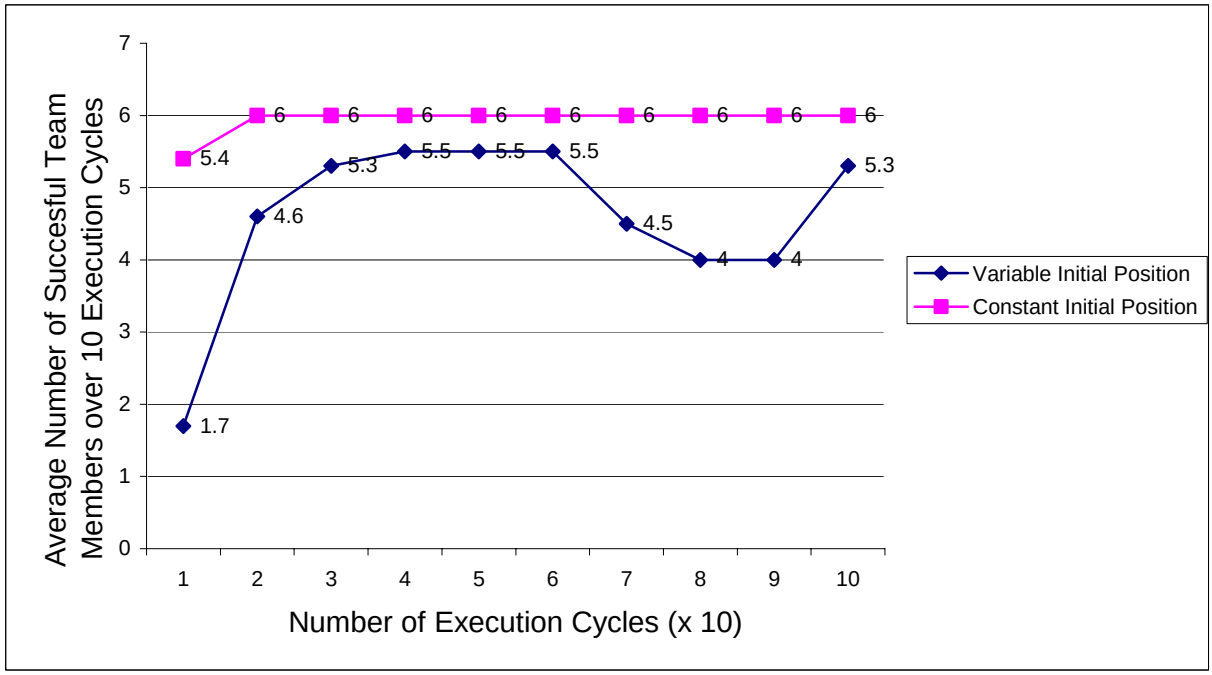

Figure 14. The Effect of Uncertainty due to the Initial Robot Positioning

7.5 Experimental Results

In order to verify the validity of the social network based approach for task allocation, various experiments were performed. When interpreting the experimental results, various aspects of the social network based approach were considered, ranging from learning to interpretation of the results in the light of social sciences.

The effects of uncertainty are visible in the results presented in this section. Even when the team is stable (consisting of the same team members), there are some fluctuations in the results. The fluctuations are there because of the introduced uncertainty, as described in section 7.4.

The experiments are divided into the following categories:

- Experiments that compare the social networks based approach to an auction based approach in relation to a single environment type.

- Experiments that compare the social networks based approach to an auction based approach in relation to multiple environment types.
- Experiments that explore the specialisation ability of robots with regard to a particular task.
- Experiments that explore learning capabilities of the social networks based approach to coordination.
- Experiments that explore the influence of changes in ratio between trust, kinship and history on the selection process.
- Experiments that explore the effects of probabilistic selection on the overall performance.
- Experiment that investigates formation of subgroups within a social network.

In all of the above enumerated simulations the same population of fifty randomly-created agents was used. The population, with its attribute values, is given in table 9. Each row represents one of the agents in the population.

LOAD	DRIVE	SPEED	DETECTION	AUTONOMY	FOODSENSOR
SMALL	WHEEL	FAST	NORMAL	TETHERED	YES
NORMAL	WHEEL	LOW	LIGHT ONLY	SOLAR ONLY	NO
SMALL	LEGS	FAST	LIGHT ONLY	SOLAR ONLY	NO
SMALL	WHEEL	FAST	NORMAL	SOLAR ONLY	YES
SMALL	TRACK	LOW	LIGHT ONLY	BATTERY	YES
NORMAL	TRACK	LOW	LIGHT ONLY	SOLAR ONLY	YES
NONE	TRACK	LOW	LIGHT ONLY	TETHERED	YES
NONE	WHEEL	LOW	NORMAL	BATTERY	YES
NONE	LEGS	MEDIUM	NORMAL	SOLAR ONLY	NO
NORMAL	TRACK	FAST	NORMAL	BATTERY	NO
NONE	LEGS	FAST	LIGHT ONLY	SOLAR ONLY	NO
NONE	WHEEL	MEDIUM	LONG RANGE	TETHERED	YES
NORMAL	LEGS	FAST	LONG RANGE	BATTERY	NO
NONE	LEGS	LOW	LIGHT ONLY	TETHERED	YES
NORMAL	LEGS	FAST	NORMAL	BATTERY	YES
NORMAL	WHEEL	MEDIUM	NORMAL	BATTERY	NO
NONE	WHEEL	FAST	LONG RANGE	SOLAR ONLY	NO
NONE	TRACK	LOW	LONG RANGE	TETHERED	NO
NORMAL	TRACK	LOW	LONG RANGE	TETHERED	YES
SMALL	LEGS	FAST	LONG RANGE	TETHERED	NO
NONE	LEGS	FAST	NORMAL	BATTERY	NO
NONE	LEGS	MEDIUM	LIGHT ONLY	SOLAR ONLY	NO
NORMAL	LEGS	MEDIUM	NORMAL	SOLAR ONLY	YES
NORMAL	LEGS	FAST	NORMAL	BATTERY	YES

NONE	TRACK	LOW	LONG RANGE	BATTERY	NO
NONE	TRACK	MEDIUM	LONG RANGE	BATTERY	NO
SMALL	WHEEL	FAST	NORMAL	BATTERY	NO
NONE	LEGS	LOW	LONG RANGE	BATTERY	YES
NONE	TRACK	MEDIUM	NORMAL	BATTERY	YES
NONE	WHEEL	LOW	NORMAL	SOLAR ONLY	YES
NONE	WHEEL	MEDIUM	LIGHT ONLY	BATTERY	NO
NONE	LEGS	MEDIUM	NORMAL	SOLAR ONLY	YES
NORMAL	WHEEL	FAST	NORMAL	BATTERY	NO
NORMAL	TRACK	FAST	LONG RANGE	TETHERED	YES
NONE	TRACK	LOW	NORMAL	TETHERED	YES
NORMAL	WHEEL	MEDIUM	LIGHT ONLY	SOLAR ONLY	NO
SMALL	WHEEL	LOW	NORMAL	TETHERED	YES
SMALL	TRACK	LOW	LONG RANGE	BATTERY	YES
SMALL	WHEEL	LOW	NORMAL	SOLAR ONLY	YES
NORMAL	TRACK	LOW	LIGHT ONLY	BATTERY	YES
NONE	WHEEL	FAST	LONG RANGE	SOLAR ONLY	YES
NORMAL	WHEEL	LOW	NORMAL	BATTERY	NO
SMALL	WHEEL	FAST	LIGHT ONLY	BATTERY	NO
NONE	LEGS	FAST	LONG RANGE	SOLAR ONLY	NO
NONE	WHEEL	MEDIUM	LIGHT ONLY	SOLAR ONLY	NO
NONE	WHEEL	MEDIUM	NORMAL	TETHERED	YES
NONE	TRACK	MEDIUM	LIGHT ONLY	TETHERED	NO
NORMAL	WHEEL	MEDIUM	LIGHT ONLY	BATTERY	YES
NORMAL	WHEEL	MEDIUM	NORMAL	BATTERY	YES
NORMAL	LEGS	LOW	NORMAL	BATTERY	YES

Table 9. Agent population created and used for experiments in this chapter

Each experiment used the same agent population. Each experiment was done using a simulation that consists of a number of execution cycles, usually 100 execution cycles (unless stated otherwise). An execution cycle consists of execution of a scouting task and a foraging task. Each of the tasks in turn consists of the five cooperative problem-solving cycle components as given in section 5.5.4.

It is important to note that each execution cycle can be seen as a small-scale simulation, as it independently simulates a complete task execution. Each execution cycle is subject to uncertainty due to the initial robot positioning (refer to section 7.4.2) and usually to uncertainty due to environment variations (refer to section 7.4.1). Even if the environments are of the same type, they are randomly created with random positioning of food items and obstacles. However, the execution cycles are not totally independent simulations. Historical performance, as embedded in trust relationships, is passed from one execution cycle to the next one. In other words,

through social relationships, the society maintains a performance history for each agent.

For each experiment, the simulation was defined using a set of parameters. The default simulation parameters are given in table 10. In all simulations, initial trust between agents is set to zero. In other words, there is no history between the agents. In the case of a scouting task, there is only one agent who executes the scouting task. In the case of the foraging task, the default (and maximum) team size is six members. The maximum number of successful team members is therefore six. The default environment type is variable, in other words the simulation is not restricted to a single environment type.

Simulation Parameters	Default Values
Execution Cycle	Scouting, Foraging
Number of Execution Cycles	100
Environment Type	Variable
Random Initial Positioning	Yes
Foraging Team Size	6

Table 10. Default simulation parameters

The results of simulations are presented and discussed next. It is important to note that the data points in figures 14-21 represent the average value over 10 execution cycles.

7.5.1 Performance Comparison to an Auction Based Approach (Single Environment Type)

For the first experiment, the performance of the new social networks based approach to task allocation was compared to the performance of a simple auctioning mechanism. Only uncertainty due to environment variations was introduced in this simulation. The same environment type is used. Table 11 provides a summary of the simulation parameters.

Simulation Parameters	Default Values
Execution Cycle	Scouting, Foraging
Number of Execution Cycles	200 – 300
Environment Type	Single
Random Initial Positioning	Yes
Foraging Team Size	6

Table 11. Default simulation parameter for a performance comparison to an auction based approach (single environment) simulation.

The auctioning mechanism is aware of all ENVIRONMENT_DETAILS attribute values, with the exception of the LIGHT attribute. The auctioning mechanism selects agents according to environment attribute and rules, as described section 7.1. Uncertainty is simulated by omitting the LIGHT attribute. Figure 15 gives the results after 200 execution cycles.

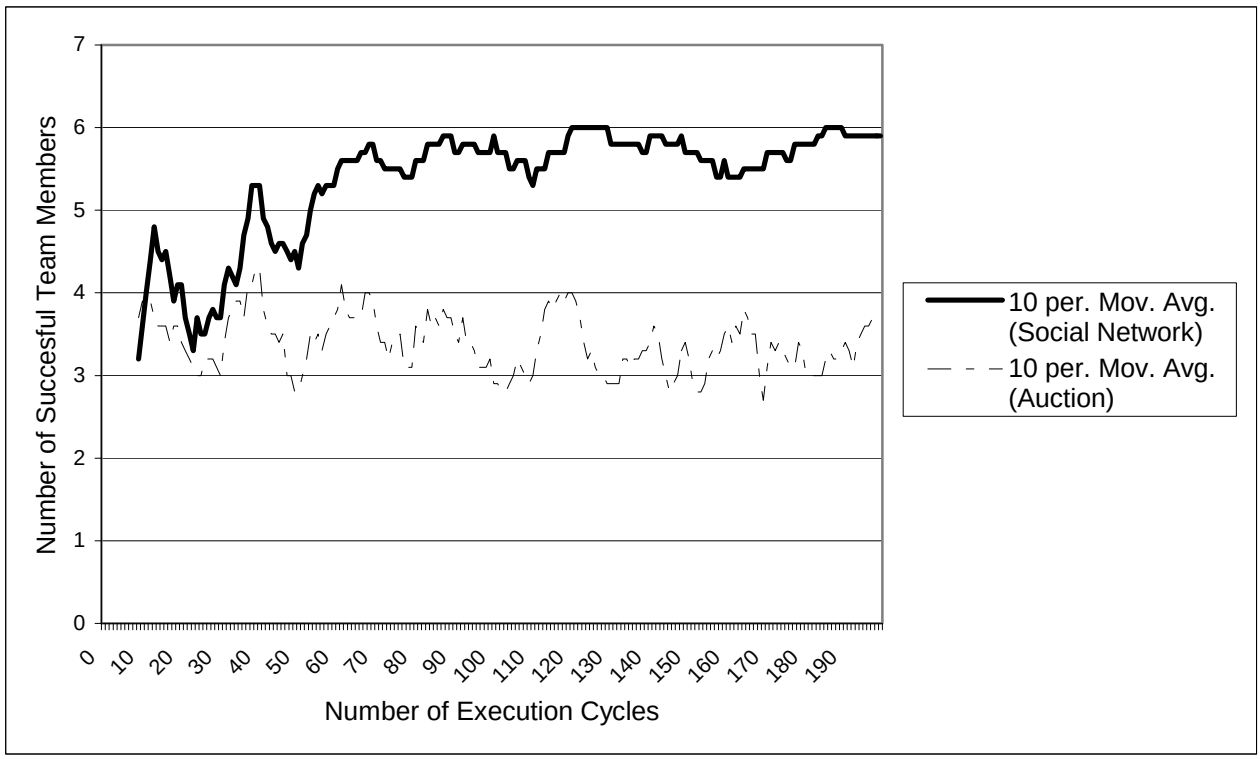

Figure 15. Performance comparison between social networks based approach and auctioning approach on single environment

The main difference between the simple auctioning and the social networks based approach is in the prior knowledge used. The social networks based approach has no prior knowledge and performance is initially less than the performance of the auctioning mechanism. The experiments showed that auctioning mechanism performance remains more or less stable, while the performance of the social networks approach improves over time. Social networks approach during certain execution cycles reached the optimal performance (all team members were successful). To confirm the stability of social networks approach, an additional 100 execution cycles were executed, with the results given in figure 16.

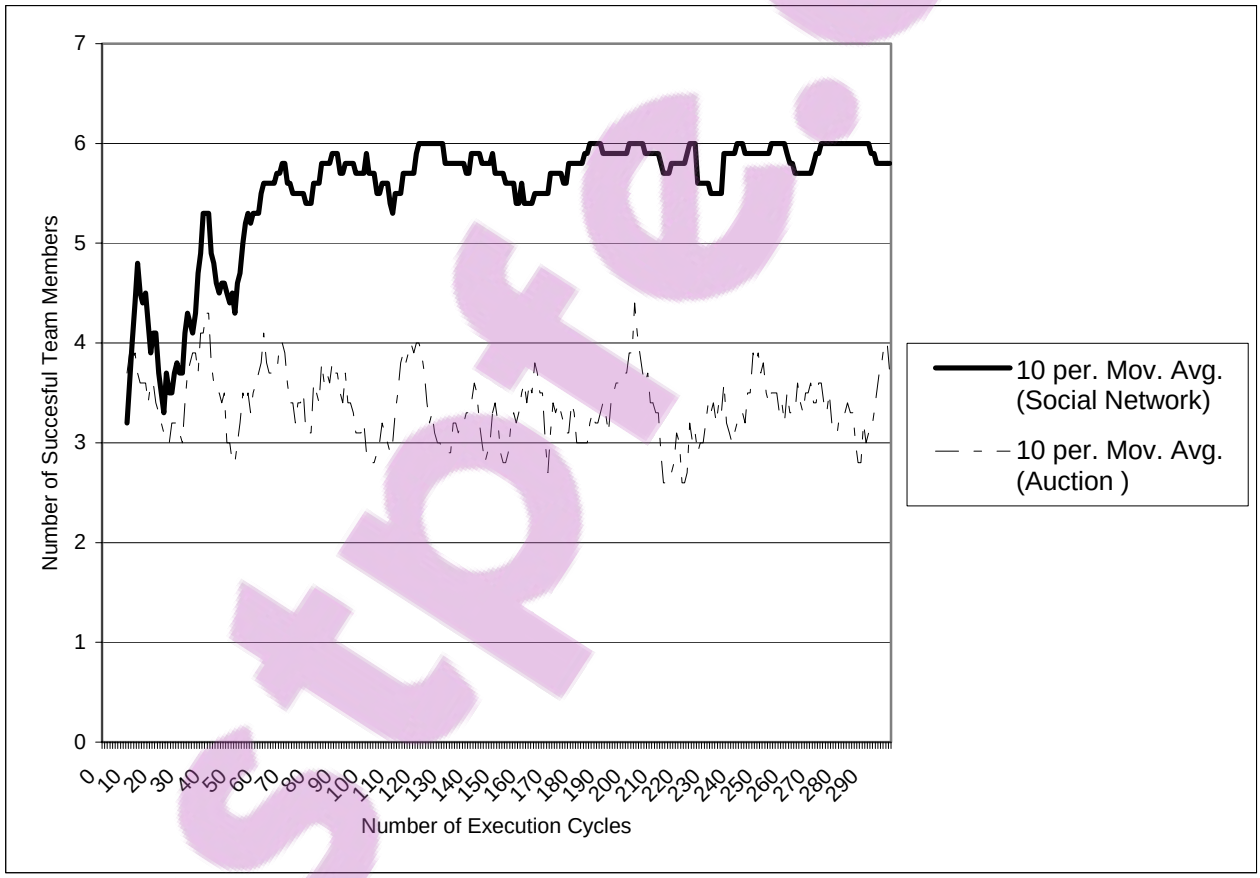

Figure 16. Performance comparison between social networks based approach and auctioning approach on single environment (300 execution cycles)

Based on these experiments, it is safe to conclude that the social networks based approach to task allocation performs significantly better than the auctioning mechanism in conditions of limited uncertainty about the task details.

7.5.2 Performance Comparison to an Auction Based Approach (Multiple Environment Types)

The results presented in section 7.5.1 are encouraging and the aim of the next experiment is to investigate if the social networks based approach performs equally well over a greater diversity of environments. In this simulation, the performance of the social network based approach to task allocation was again compared to the performance of a simple auctioning mechanism for task allocation. Uncertainty due to environment variations was introduced in this experiment (as in the previous experiment, refer to section 7.5.1). It is important to note that environments change from execution cycle to execution cycle. For each execution cycle, an environment was created by randomly choosing its attribute values. The aim of the simulation is to investigate if the social networks based approach generalises and if it can handle different environment types. A summary of simulation parameters is given in table 12.

Simulation Parameters	Default Values
Execution Cycle	Scouting, Foraging
Number of Execution Cycles	200 – 300
Environment Type	Variable
Random Initial Positioning	Yes
Foraging Team Size	6

Table 12. Parameters for a performance comparison to an auction based approach (multiple environment types) simulation.

The implementation of the auctioning mechanism is the same as described in section 7.5.1. The results after 200 execution cycles are illustrated in figure 17.

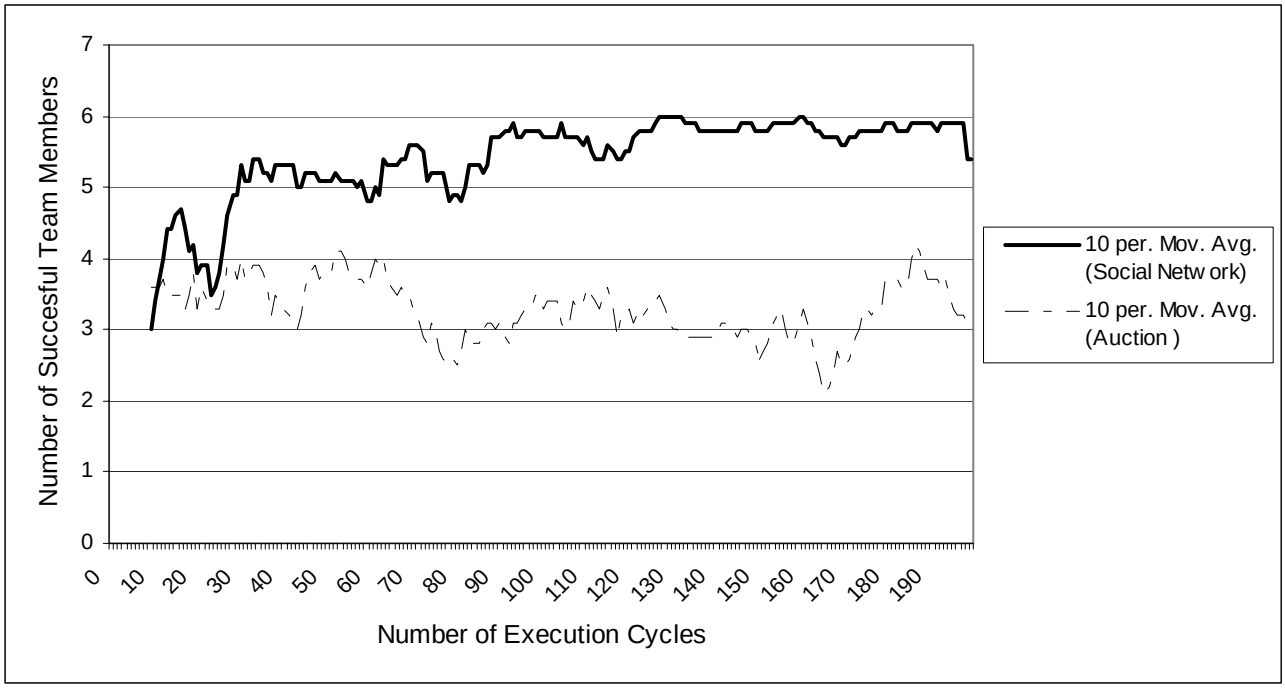

Figure 17. Performance comparison between social networks based approach and auctioning approach on multiple environments (200 execution cycles)

The drop in performance in the last ten execution cycles (refer to figure 17) warranted further investigation and the number of execution cycles was increased to 300 in order to check if it was just a random fluctuation or if it was an indication of a downward trend. As can be seen in figure 18, it was just a random fluctuation.

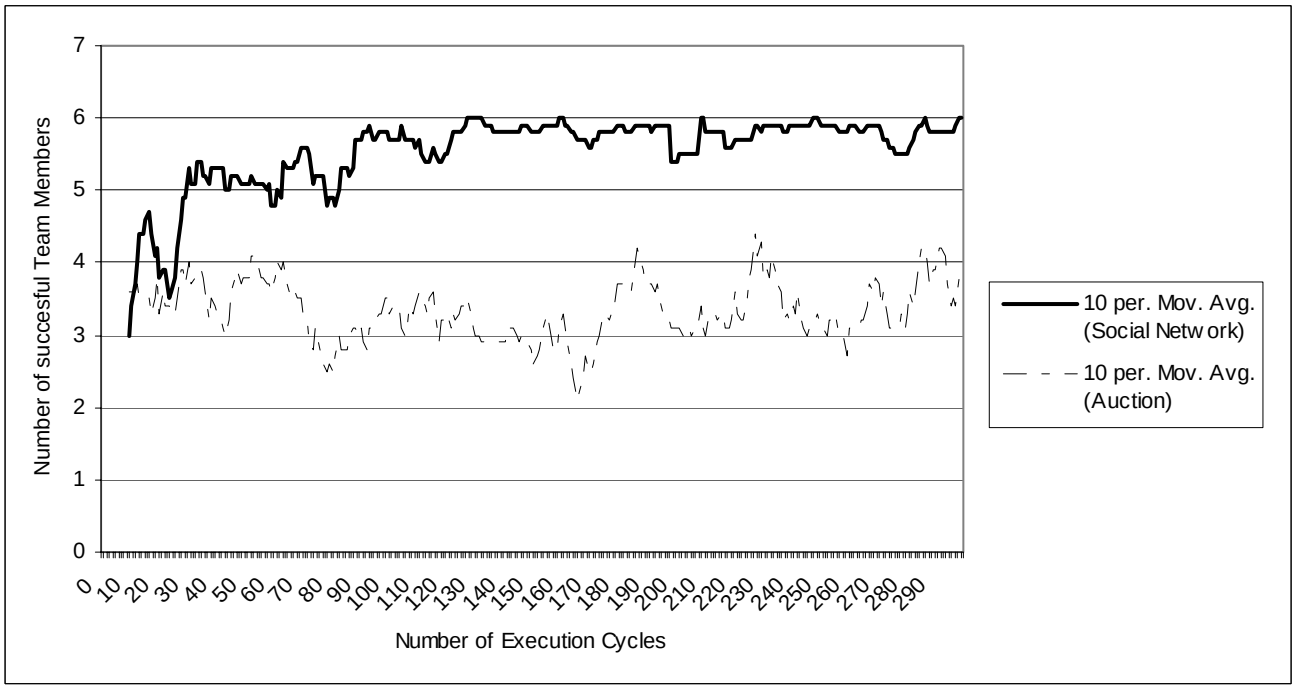

Figure 18. Performance comparison between social networks based approach and auctioning approach on multiple environments (300 execution cycles)

The observed performance leads to the conclusion that the social networks based approach to task allocation performs significantly better than auctioning mechanism in conditions of limited uncertainty over the task details and over multiple environment types.

7.5.3 The Influence of Probabilistic Selection

The social networks based approach uses a straightforward team selection method: the members with the highest scores (bids) are selected. As this is not the only possible team selection method, this section investigates the performance of different team selection methods for the social networks based approach. A probabilistic selection, based on roulette wheel selection, was implemented and compared with standard rank-based selection. Uncertainties due to environment variations and initial positioning were introduced in this experiment. The simulation was executed for 200 execution cycles over single and multiple environment types. A summary of the simulation parameters is given in table 13.

Simulation Parameters	Default Values
Execution Cycle	Scouting, Foraging
Number of Execution Cycles	200
Environment Type	Single, Variable
Random Initial Positioning	Yes
Foraging Team Size	6

Table 13. Simulation parameters used for the investigation of probabilistic selection influence.

Figures 19 and 20 respectively present the results for a single environment and for multiple environments.

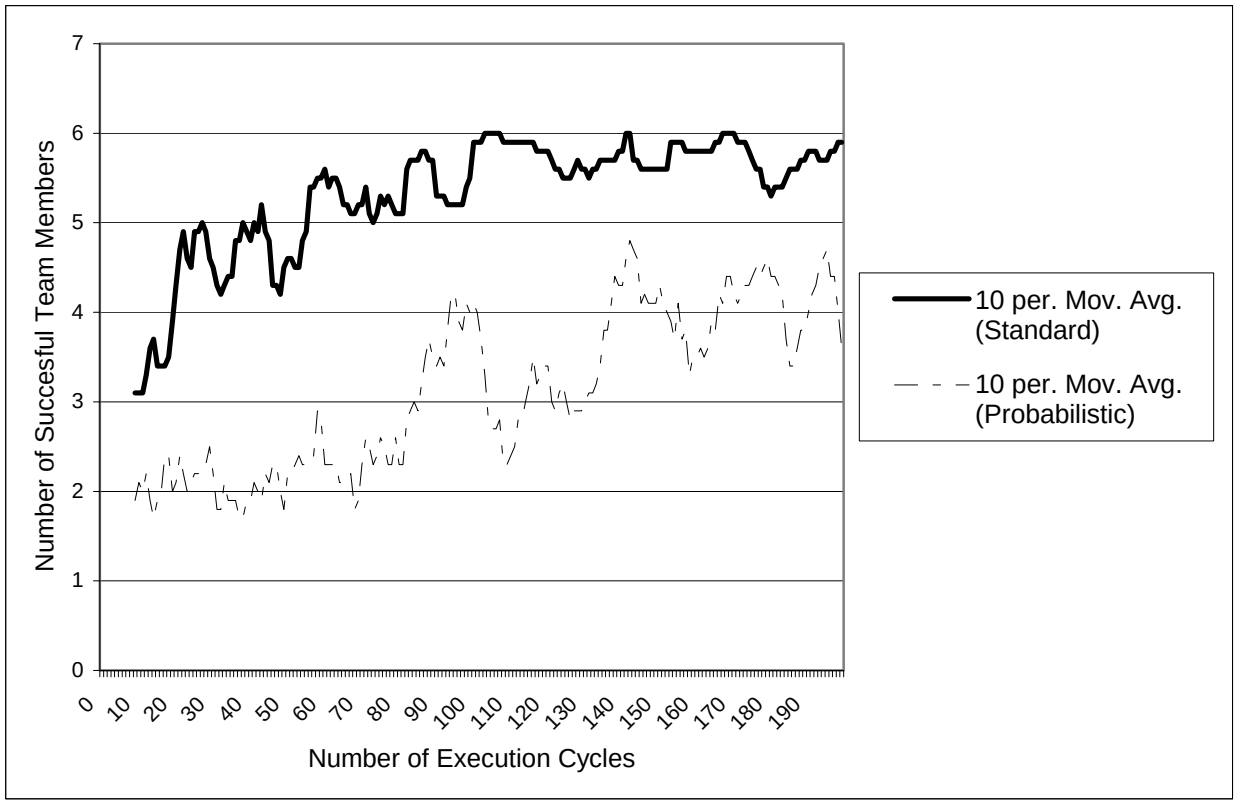

Figure 19. Performance comparison between standard and probabilistic selection on single type environment (200 execution cycles)

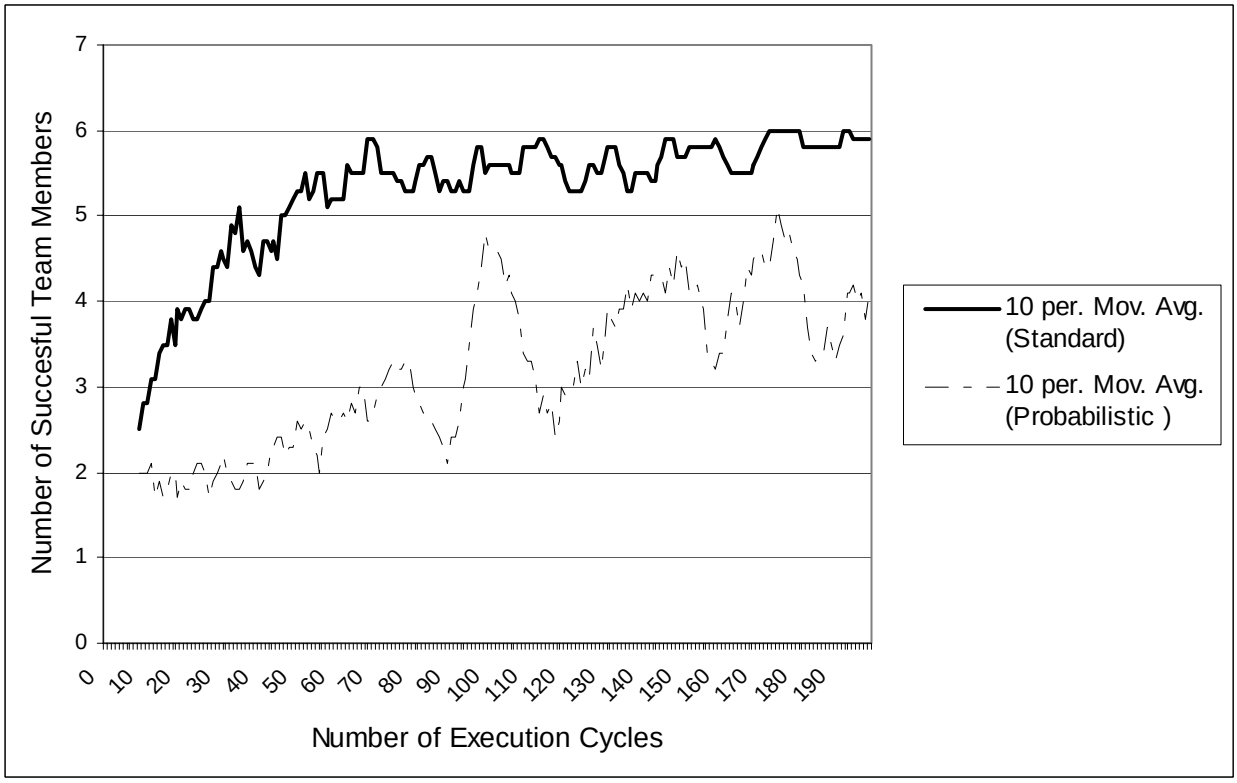

Figure 20. Performance comparison between standard and probabilistic selection over multiple environment types (200 execution cycles)

It is important to note that, although slower than standard rank-based selection, the performance using the probabilistic selection also steadily improves over time. The advantage of probabilistic selection is in the fact that more agents get tested over time for their suitability to a task. This approach might be feasible for future applications that allow for a greater training period and where the availability of agents might be scarce, but in environments were all agents are available and there is a time constraint, the standard approach is more desirable.

7.5.4 Learning Using Social Networks Approach

One of the most useful features of social networks is their ability to store information [179]. The experiments presented in this section concentrate on increasing the number of execution cycles while keeping uncertainty limited to the randomness in the initial positions of the robots. Two simulations were done to investigate the ability of social networks to learn. For the purpose of these experiments, the ability to improve on performance is seen as a learning ability. The performance is defined as the number of successful team members. For the first experiment, a single environment was used, while the second experiment used variable environments. The results are presented and discussed next.

7.5.4.1 Learning over Single Environment

For the simulation in the first experiment, the number of execution cycles has increased in steps of 10, from 0 to 100. Only uncertainty in this experiment is due to initial positioning. A summary of the simulation parameters is given in table 14.

Simulation Parameters	Default Values
Execution Cycle	Scouting, Foraging
Number of Execution Cycles	100
Environment Type	Single
Random Initial Positioning	Yes
Foraging Team Size	6

Table 14. Simulation parameters used for the investigation of probabilistic selection influence

The results are presented in figure 21. The results show that the social networks approach improves performance over time. It is important to note that optimal performance is reached after 8 execution cycles. As previously noted, the improvement in performance is seen as learning ability. The learning characteristic of social networks has been observed in organisations [102].

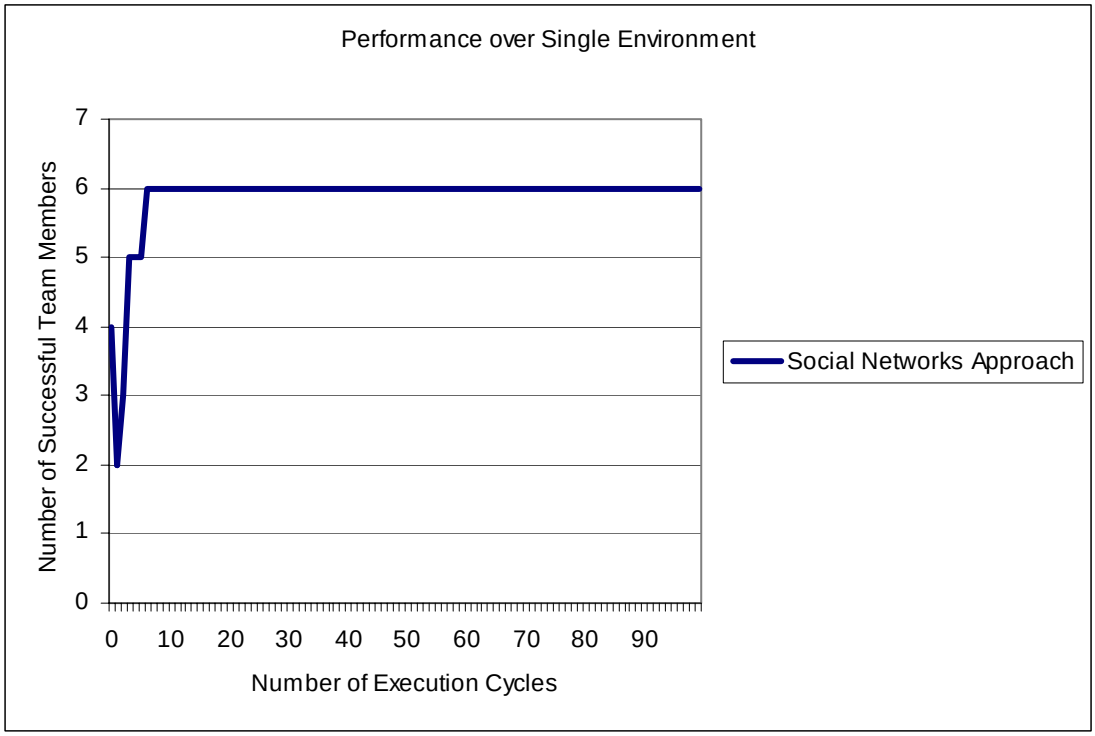

Figure 21. Observed improvement in performance (single environment)

7.5.4.2 Learning over Variable Environments

For this simulation, the influence of variable environment types on learning was investigated. As in previous experiment, uncertainty in this experiment is due to initial positioning. A summary of the simulation parameters is given in table 15.

Simulation Parameters	Default Values
Execution Cycle	Scouting, Foraging
Number of Execution Cycles	200
Environment Type	Variable
Random Initial Positioning	Yes
Foraging Team Size	6

Table 15. Simulation parameters used for the investigation of probabilistic selection influence

The results are presented in figure 22. Initially, after 100 execution cycles, the performance was not as good as for the previous experiment (refer to section 7.5.4.2). The number of execution cycles was then increased to 200, after which performance was approached the optimal performance.

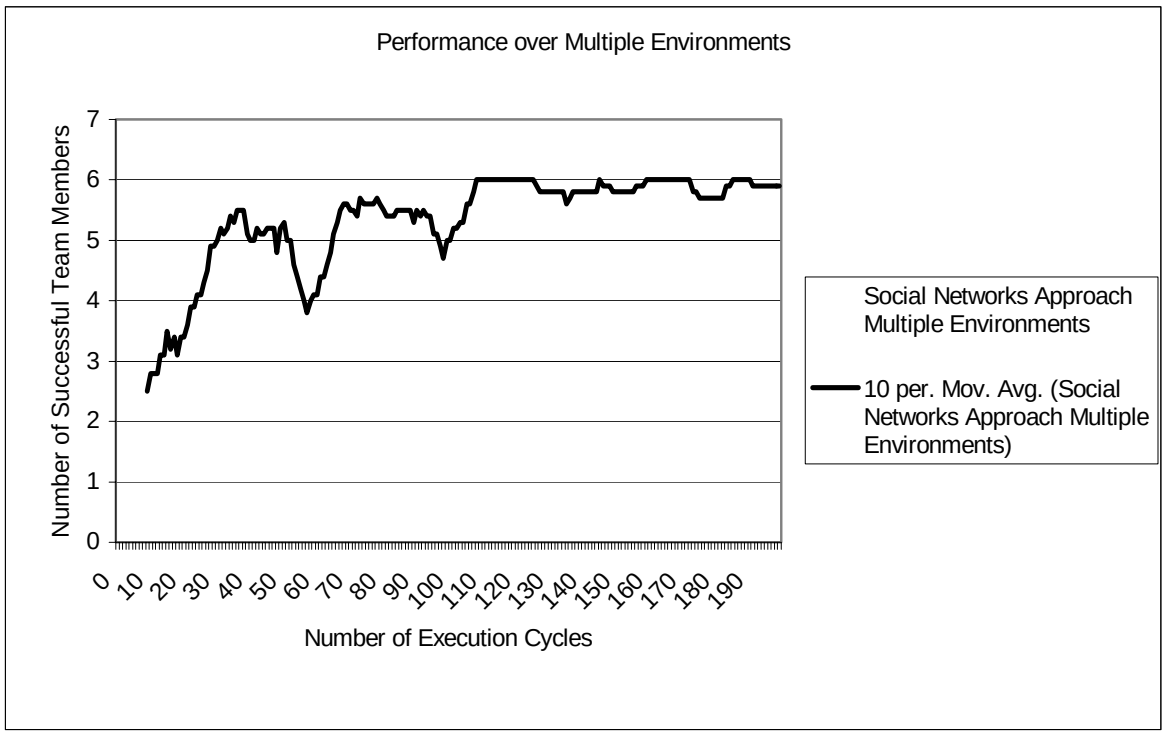

Figure 22. Observed improvement in performance (multiple environments)

Learning in multi-robot teams is a highly desirable feature due to the inherent challenges of real-world robotic applications. Some of the challenges facing real-world multi-robot teams are: uncertainty in sensing the environment (due to imperfect sensors), limited amount of historical performance (for training purposes) and difficulties in non-symbolic learning mechanisms, to name but a few. More on multi-robot team learning can be found in [143].

7.5.5 Agent Specialisation

The set of simulations presented in this section concentrate on the effect of social networks on agent specialisation. Specialisation is often observed in learning capable multi-robot architectures. For example, Balch reports that in such, learning capable, multi-robot team, individual robots automatically specialise in different roles in a team [13].

For the purpose of this paper, only one scout was allowed to assume the task of exploring the environment. The environment had the following attributes: TERRAIN_NORMAL, SHADED_AREAS, FOOD_FAR, FOOD_HEAVY. The success of the scout was measured by the ability of the scout to find the food area in a given number of steps. It is interesting to observe how various robots first attempted the scout role but in the end only one robot emerged as the best scout. Initially, any robot can be selected for either scouting or foraging tasks. If the robot is successful in the scouting task, the robot is kept as a scout. If a robot's performance in executing the scout task is less then an arbitrary threshold (defined as the ratio between the successful and attempted number of task executions), it is replaced by another robot. For the purpose of this thesis, the threshold was set to 0.8. The threshold can be interpreted as a minimum performance criterion. In other words, a scout with a success rate greater than 80% is an acceptable scout. The simulation executed for 100 execution cycles, although stability was reached after 48 execution cycles. A summary of the simulation parameters is given in table 16.

Simulation Parameters	Default Values
Execution Cycle	Scouting
Number of Execution Cycles	100
Environment Type	Variable
Random Initial Positioning	Yes
Foraging Team Size	N/A

Table 16. Simulation parameters used for the investigation of probabilistic selection influence

After 48 execution cycles, one of the agents, characterised by the attribute value vector (SMALL, LEGS, FAST, LONG RANGE, BATTERY, NO) (refer to table 7) emerged as the best scout. Table 17 illustrates the attribute values of the 24 agents that were considered for the scouting task. The first column represents the number of execution cycles (out of total of 100) that an agent was selected for the scouting task.

TRIES	LOAD	DRIVE	SPEED	DETECTION	POWER	FALSE FOOD
1	NONE	WHEEL	MEDIUM	LIGHT ONLY	SOLAR	NO
1	SMALL	TRACK	LOW	LONG RANGE	TETHERED	YES
1	NORMAL	LEGS	LOW	LONG RANGE	BATTERY	NO
3	NONE	TRACK	FAST	LONG RANGE	SOLAR	YES
1	NONE	LEGS	FAST	LONG RANGE	SOLAR	YES
1	NONE	WHEEL	FAST	LIGHT ONLY	SOLAR	NO
1	SMALL	WHEEL	FAST	NORMAL	TETHERED	YES
1	NONE	LEGS	FAST	LONG RANGE	SOLAR	NO
1	SMALL	WHEEL	LOW	NORMAL	TETHERED	YES
22	NORMAL	WHEEL	MEDIUM	NORMAL	BATTERY	YES
1	NONE	TRACK	FAST	LIGHT ONLY	SOLAR	NO
2	NONE	TRACK	FAST	LONG RANGE	SOLAR	NO
1	NORMAL	LEGS	LOW	LIGHT ONLY	TETHERED	NO
1	NONE	WHEEL	FAST	LIGHT ONLY	SOLAR	YES
1	NORMAL	WHEEL	FAST	NORMAL	BATTERY	NO
2	NORMAL	TRACK	LOW	LONG RANGE	SOLAR	NO
1	NONE	TRACK	MEDIUM	LONG RANGE	TETHERED	NO
1	SMALL	TRACK	MEDIUM	LIGHT ONLY	BATTERY	YES
1	NONE	TRACK	MEDIUM	NORMAL	BATTERY	YES
1	NORMAL	TRACK	LOW	LONG RANGE	TETHERED	NO
1	SMALL	LEGS	FAST	LIGHT ONLY	SOLAR	NO
1	NONE	WHEEL	MEDIUM	LONG RANGE	TETHERED	YES
1	NONE	TRACK	LOW	LONG RANGE	BATTERY	NO
52	SMALL	LEGS	FAST	LONG RANGE	BATTERY	NO

Table 17. Selected scout robot attributes

7.5.6 Influence of Kinship and Trust Parameter Values

The experiments that are presented in this section concentrate on the effects of changes in the ratio between kinship and trust. Three simulations were done with different parameter values for kinship and trust, while the rest of the simulation parameters were the same for all three simulations. A summary of the simulation parameters is given in table 18.

Simulation Parameters	Default Values
Execution Cycle	Scouting, Foraging
Number of Execution Cycles	700
Environment Type	Variable
Random Initial Positioning	Yes
Foraging Team Size	6

Table 18. Simulation parameters used for the investigation of probabilistic selection influence

The results of three sets of simulations are presented and discussed next.

7.5.6.1 Performance of The Model Using Only Kinship

For this simulation, only kinship was used to calculate the strength of social networks. Referring to equation 7.2, the value of parameter k is 1. The results are presented in figure 23. The result was somewhat surprising, as the social networks based approach still demonstrated the ability to learn by improving its performance over time. Trust was envisaged to be the main mechanism that stores historical performance, so how was it possible to improve performance over the time? The answer lies in the fact that the strength of kinship is calculated in relation to the scout. This means that as scout selection improves, the performance of the whole team improves. The rest of the team is now selected according to the kinship relationship to a scout that is better than previous scouts. A scout is better than another scout if its attributes are more suited to a variety of environments. By selecting the team members that share similar attributes (due to the strong kinship), the performance of team improves, however the maximum performance is not reached.

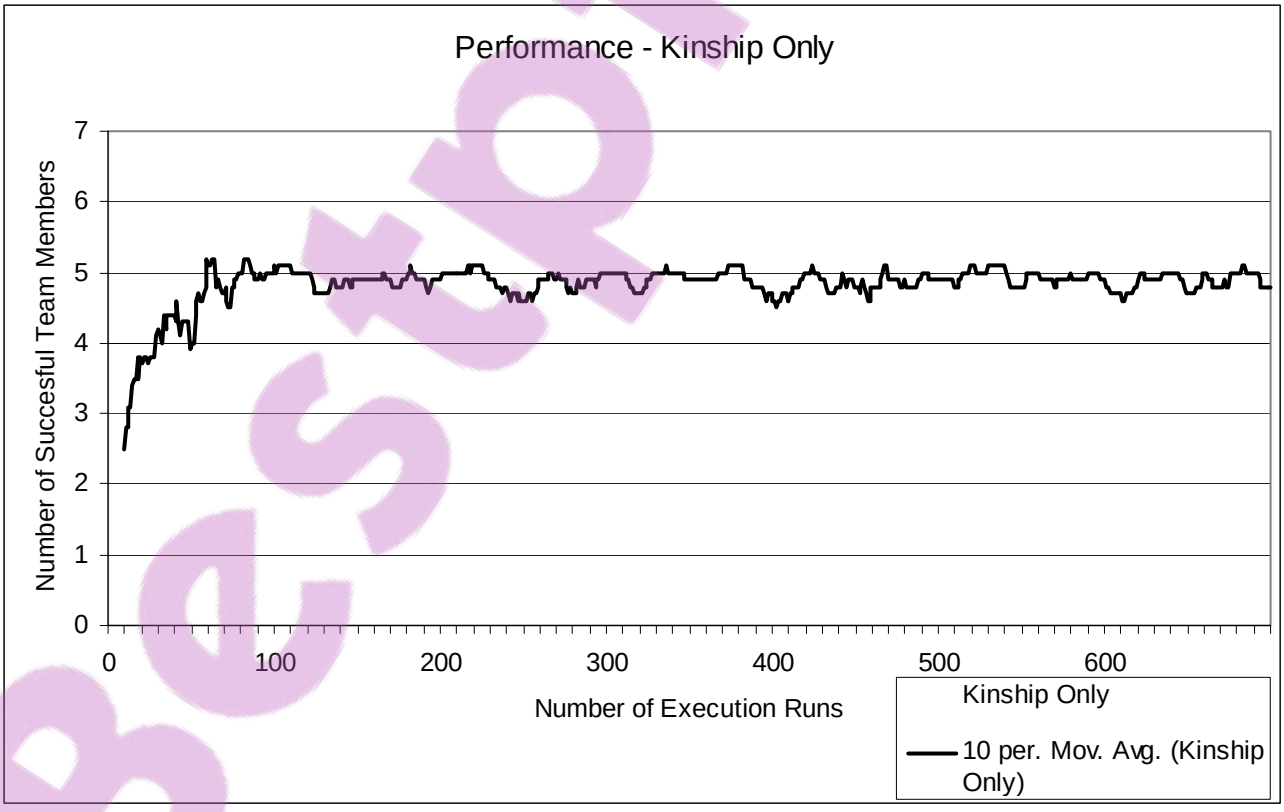

Figure 23. Performance of social networks based approach with only kinship relationship (700 execution cycles)

7.5.6.2 Performance of The Model Using Only Trust

For this experiment, only trust was used in the calculation of the strength of social networks. Referring to equation 7.2, the value of parameter k is 0.

The results are presented in figure 24. As for the previous experiment, the social networks based approach demonstrated the ability to learn by improving its performance over time. In the social networks based approach, the trust is the main mechanism that allows for storage of the historical data that reflects the past performance of team members. It is also important to note that maximum performance is also occasionally reached.

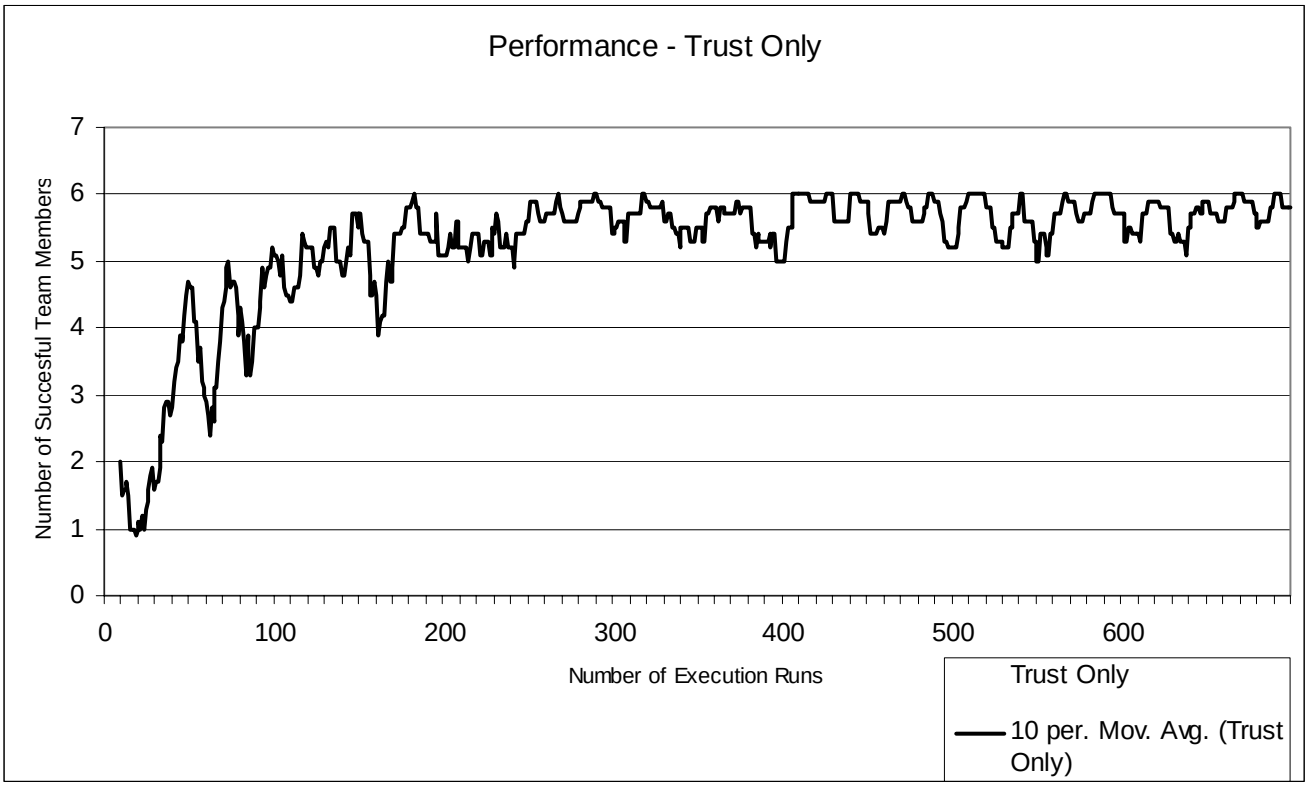

Figure 24. Performance of social networks based approach with one social relationship only – trust relationship (700 execution cycles)

7.5.6.3 Performance of The Model Using Trust and Kinship

In the last experiment that explores the effect of various parameter values to the performance of the social networks based approach the value of parameter k in equation 7.2 was set to 0.3. Various values ranging from 0.1 to 0.7, in increments of 0.1, have been tested and the best performance has been observed for the value of 0.3.

The results are presented in figure 25. The results were similar to the results from previous experiments. The ability of social networks to improve their performance over the time has been demonstrated in all experiments conducted for the purpose of this thesis.

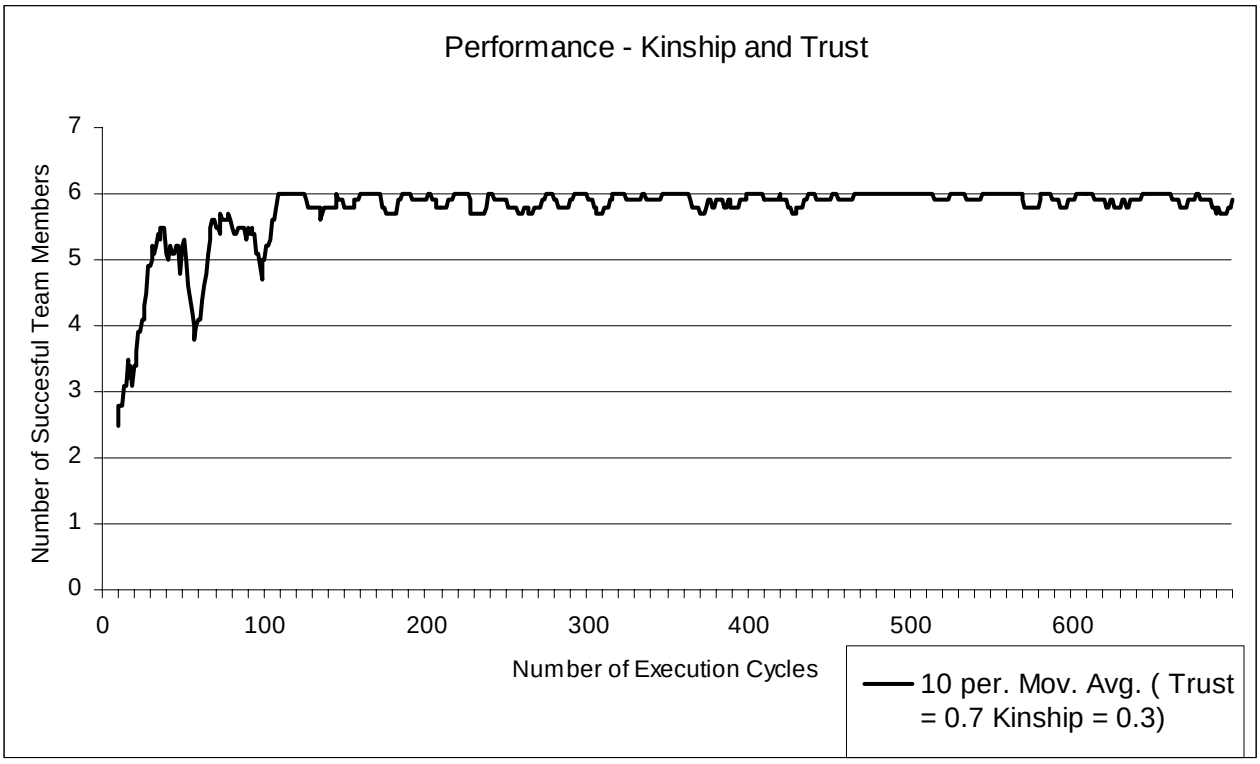

Figure 25. Performance of social networks based approach with one social relationship only – kinship relationship (700 execution cycles)

7.5.6.4 Discussion on Effects of Different Parameter Values

The social networks based approach to team allocation has demonstrated the ability to improve performance regardless of the value of the parameters that determine the

ratio between kinship and trust (refer to equation 7.2). The results presented in the previous sections do, however, show that the behaviour of the social networks based approach differs for different values of parameter k.

The results of experiments with varying value of k that were presented in sections 7.5.6.1 – 7.5.6.3 are combined in figure 26.

Initially, while there is no trust relationship (in other words, no historical performance data), the kinship relationship plays the predominant role in determining the strength of social networks. To illustrate this characteristic, the results of the first 100 execution cycles for all three values of k are given in table 19. A column represents number of successful executions per ten execution cycles. Maximum is 60, as there are six team members and ten execution cycles.

Execution Cycle	Successful Executions Kinship Only	Successful Executions Trust Only	Successful Executions Trust = 0.7 Kinship = 0.3
10	23	18	23
20	35	9	31
30	41	19	49
40	44	27	55
50	43	45	48
60	48	31	40
70	47	38	56
80	50	42	57
90	49	50	55
100	50	49	49

Table 19. Comparison of social networks over first 100 execution cycles

It is interesting to note that a social network that uses both trust and kinship relationship performs either as the best approach or the second best. In other words, the performance of the combined approach is never the worst. This characteristic is observed throughout the experiments, leading to conclusion that the combined trust and kinship approach is the safest, if not always the optimal, option.

As historical data based on previous execution cycles grows, the performance of all three social networks improve over time. This is illustrated in figure 26. It is interesting to note that it takes longer for a social network that uses only the trust relationship to achieve the same level of performance as a social network that uses

only the kinship relationship, and that which uses both. However, the performance increases over time to exceed that of the kinship only method. The combined approach, using kinship and trust out-performs the single relationships approaches, albeit only after more than 500 execution cycles. The performance summary for each of the approaches after 700 execution cycles is given in table 20.

Performance over 700 execution cycles	**Kinship Only**	**Trust Only**	**Kinship and Trust**
Total number of successful team members over 700 execution cycles (max 4200)	3370	3661	4011

Table 20. Comparison of social networks over 700 execution cycles

The results of the experiments presented in sections 7.5.6.1 – 7.5.6.3 are combined in figure 26.

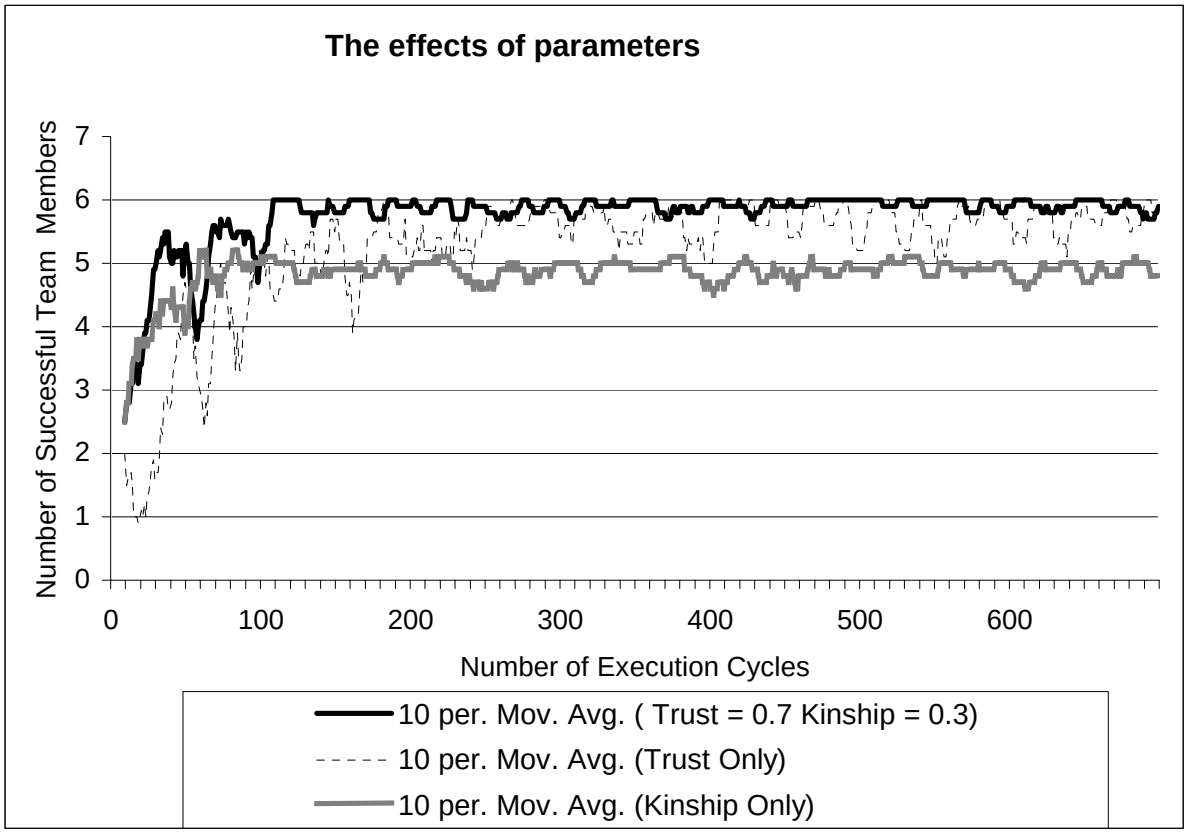

Figure 26. Performance of social networks based approach with one social relationship only – kinship relationship (700 execution cycles)

7.5.7 Evolution of Subgroups

The forming of subgroups within a social network has been observed in human societies. The forming of subgroups and their effects was studied by many researchers, mainly in the field of sociology. An overview of such research falls outside the scope of this thesis and the reader is referred to [195].

In relation to this thesis, the pertinent question is if the proposed social networks based approach indeed mimics real-life social networks as observed in higher mammalian societies. One of the characteristics of real-life social networks is the formation of social structures that are often referred to as subgroups. As noted by Collins [42], if a subgroup is tightly connected, it is referred to as a *clique*. Agents in a *clique* tend to exhibit homogenous beliefs and common characteristics.

In order to confirm that *cliques* do form between the agents in the social network based approach as presented in this thesis, this section investigates forming and evolution of such structures. In the context of an abstract simulated environment a *clique* is a subgroup of agents that are well-suited to a particular environment type. The investigation focuses on two main characteristics of a *clique* - the relative stability (once established, the members of a *clique* do not easily leave the *clique*) and the homogeneity between the members (in the context of the abstract simulator, the similarity between agents' attribute values).

The formation of *cliques* has been observed, regardless of the environment type. However, for the purpose of illustrating the process, the results presented in this section are related to one, randomly selected, environment type. The simulation used 100 execution cycles. The summary of simulation parameters is given in table 21.

Simulation Parameters	Default Values
Execution Cycle	Scouting, Foraging
Number of Execution Cycles	100
Environment Type	Single
Random Initial Positioning	Yes
Foraging Team Size	6

Table 21. Simulation parameters used for the investigation of evaluation of subgroups.

The first part of the execution cycle is the scouting task. Agent 33, defined by attribute vector (NORMAL, TRACK, FAST, LONG RANGE, TETHERED, NO), was tasked with scouting (refer to table 7 and section 7.5.5). Once the scouting task was executed, agent 33 has formed a foraging team, using the social networks based approach to team selection (refer to algorithm 10). The selected team consisted of agents 0, 12, 14, 18, 19 and 33 (refer to table 7).

During the execution of the foraging task, which is the second part of an execution cycle, four agents have completed the allocated foraging task. The graph presented in figure 27 shows the social network after the first execution cycle. Note that in order to keep the graph relatively uncluttered, the only social relationship illustrated is the trust relationship in relation to the team leader, agent 33. Nevertheless, the forming of a *clique* is clearly illustrated and the final state of the social network is given later in this section. The indices represented as a full line illustrate the current clique members, while indices represented as a dashed line indicate the agents that were considered and rejected as the clique members. The indices are weighted and their weight is given as a ratio between successful task executions and attempted task executions. The selected team leader is indicated by the capital letter T in parentheses (T).

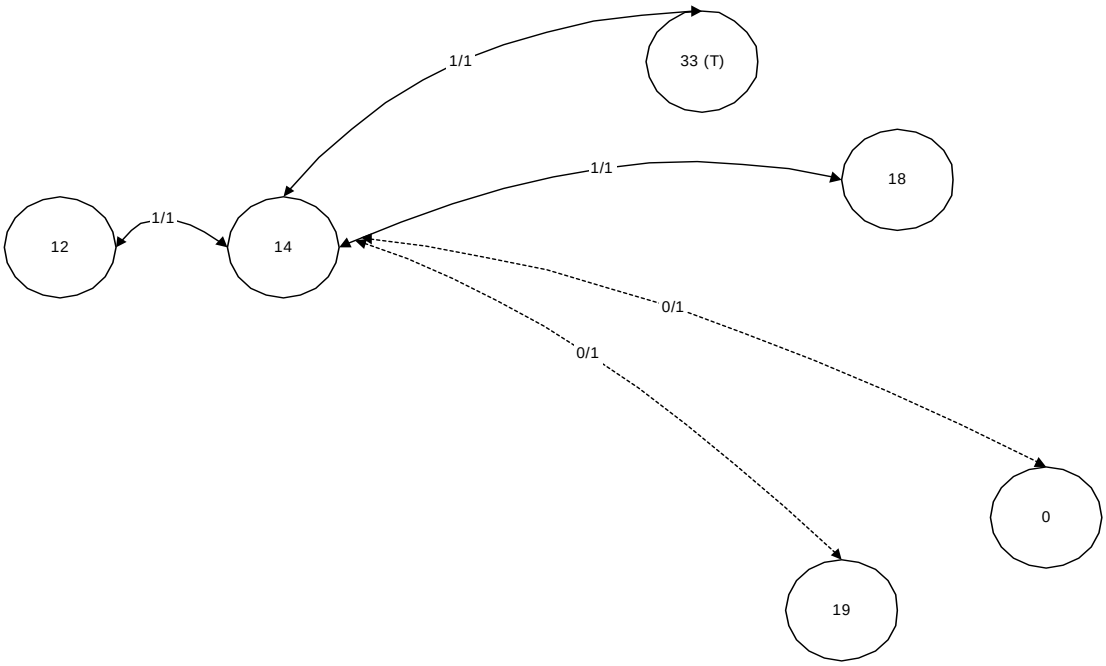

Figure 27. The sociogram after first execution cycle

Figure 28 represents the social network after the second execution cycles. During the second execution cycle, only two agents have completed the task (agents 14 and 12). The team leader, agent 33, has also failed to complete the task due to the range (agent 33 has TETHERED as attribute value of attribute POWER and according to the rules of interaction with its environment it has a limited range – refer to section 7.1). It is important to note that agents 0 and 19 have failed to complete the task due to their kinship to the scouting agent (TETHERED, refer to table 7 for agent attributes), since the scouting agent is selected as a team leader due to the initial lack of trust data. To illustrate, consider the initial state where there is no trust relationships established between the agents. The only agent with a historical performance record (agent's own trust in its suitability to perform a task in a given environment) is the scouting agent. For the kinship calculation purposes, it is considered the team leader and agents are ranked according to their kinship related to the team leader (see section 6.6.4.2).

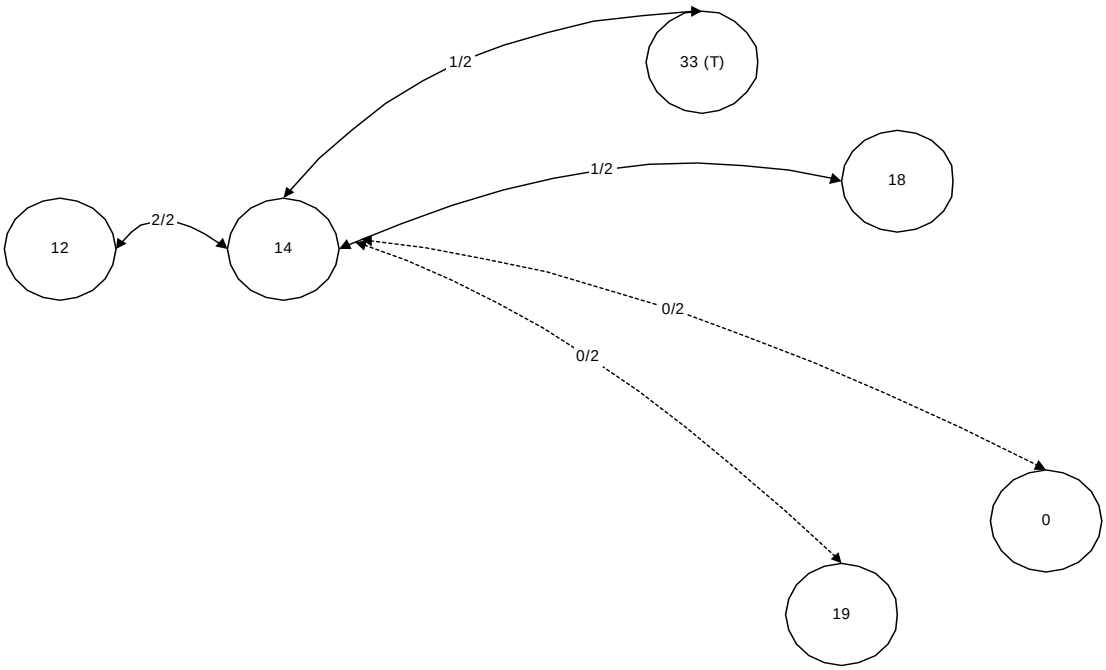

Figure 28. The sociogram after the second execution cycle

The agent's performance is considered unsatisfactory if the ratio between successful task executions and attempted task executions is less than 0.5 (a problem dependent parameter). After the second execution cycle, the performance of agent 33 that performed the roles of a scouting agent and team leader was deemed unsatisfactory. The performance of agents 0 and 19, was also deemed unsatisfactory. Agent 34 became the next scouting agent and team leader. Agents 0 and 19 were replaced by agents 49 and 23 which, were the next best ranked agents in relation to the new team leader. The foraging task was executed and agents 12, 14 and 23 have successfully completed the task. Agent 34, which was selected as the team leader, has failed to complete the task and it was replaced in the next execution cycle. Figure 29 illustrates the social network after the third execution cycle.

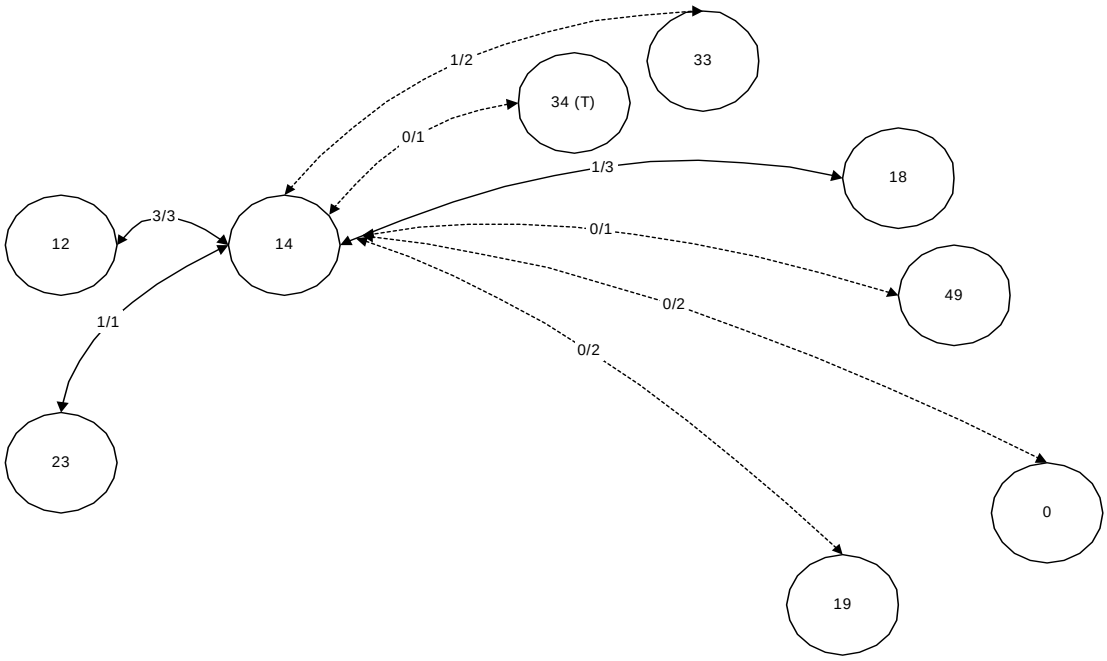

Figure 29. The sociogram after third execution cycle

It is important to note that after the third execution cycle there was sufficient historical information about previous performance related to the foraging task to enable agent 14 to become the next team leader.

As the performance of agents 18, 33 and 49 was not satisfactory, they were replaced. The next execution cycle illustrates the importance of kinship, as kinship is now calculated in relation to the new team leader. The new permanent members 9, 20, and 32, that have strong kinship relationships to the new team leader, were introduced as team members during the fourth execution cycle. The social network after the fourth execution cycle is illustrated in figure 30.

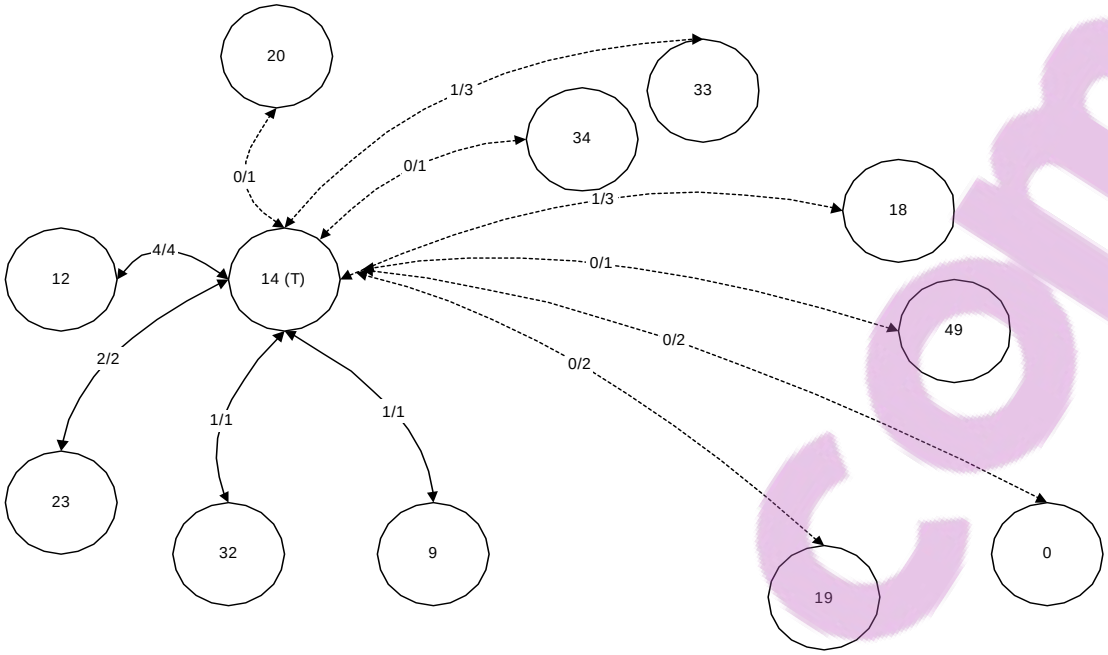

Figure 30. The sociogram after fourth execution cycle

During the fourth execution cycle all agents, with the exception of agent 20, have completed the task. Agents 12 and 14 were successful in all execution cycles so far and they can be viewed as the core of a clique. Agent 20 was replaced by a new team member, agent 43. The resulting sociogram is presented in figure 31.

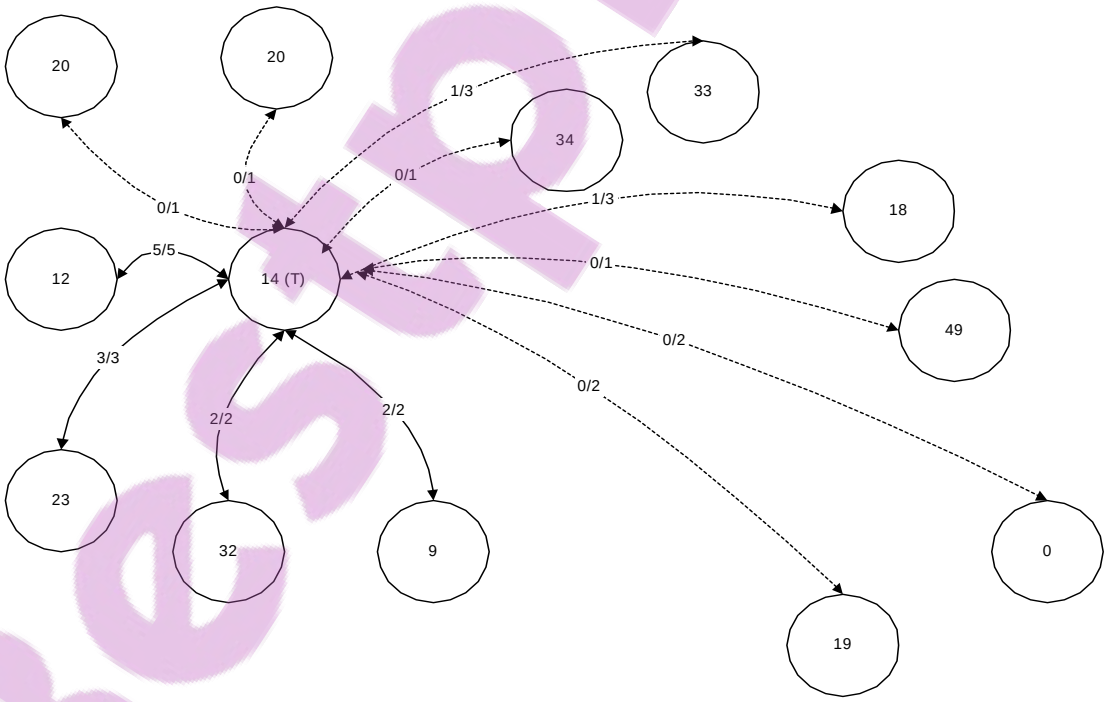

Figure 31. The sociogram after fifth execution cycle

It is important to note that after the fifth execution cycle, resilience (stability) of a clique has already been identified. Once part of the clique, members (agents 12, 23, 32 and 9) do not easily leave. The team member introduced in the fifth execution cycle, agent 43 was the only unsuccessful team member agent and agent 25 replaced it. The sixth execution cycle again further enforced the clique. The new team member, agent 25 was yet another unsuccessful team member, while all other agents have completed the task. The social network after sixth execution cycle is illustrated in figure 32.

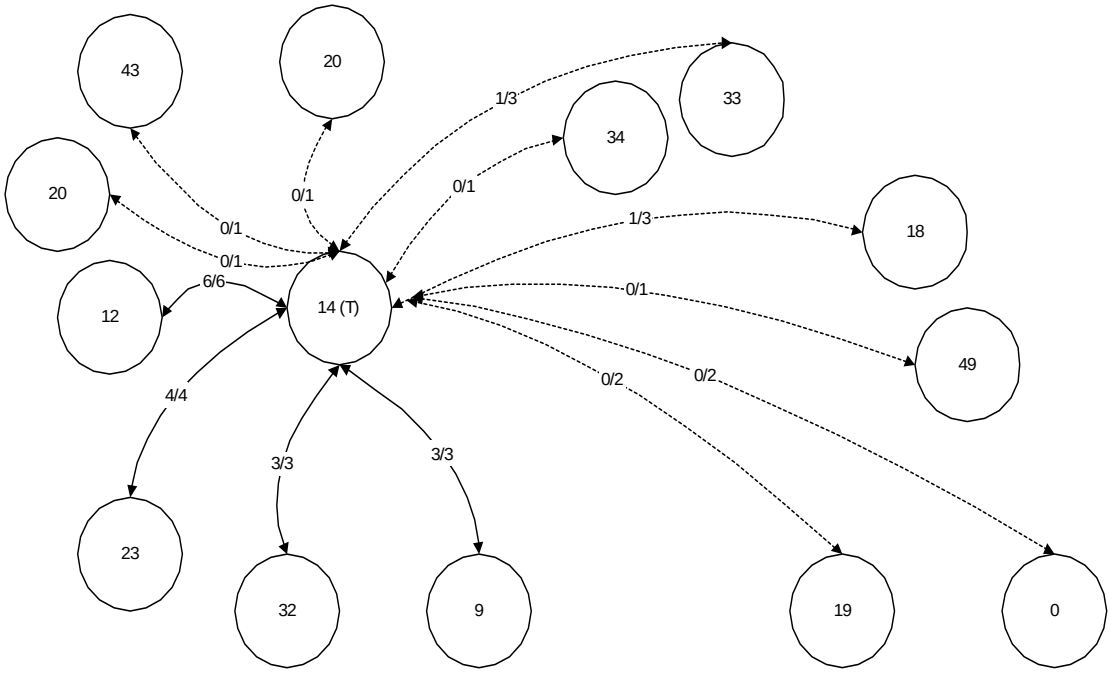

Figure 32. The sociogram after sixth execution cycle – established clique

Finally, the seventh execution cycle has introduced the sixth and final member, agent 15, of the clique around the team leader, namely agent 14. During the execution of the seventh execution cycle and subsequent execution cycles (the simulation executed 100 execution cycles) all team members were successful in task execution. The social network after the seventh execution cycle is presented in figure 33.

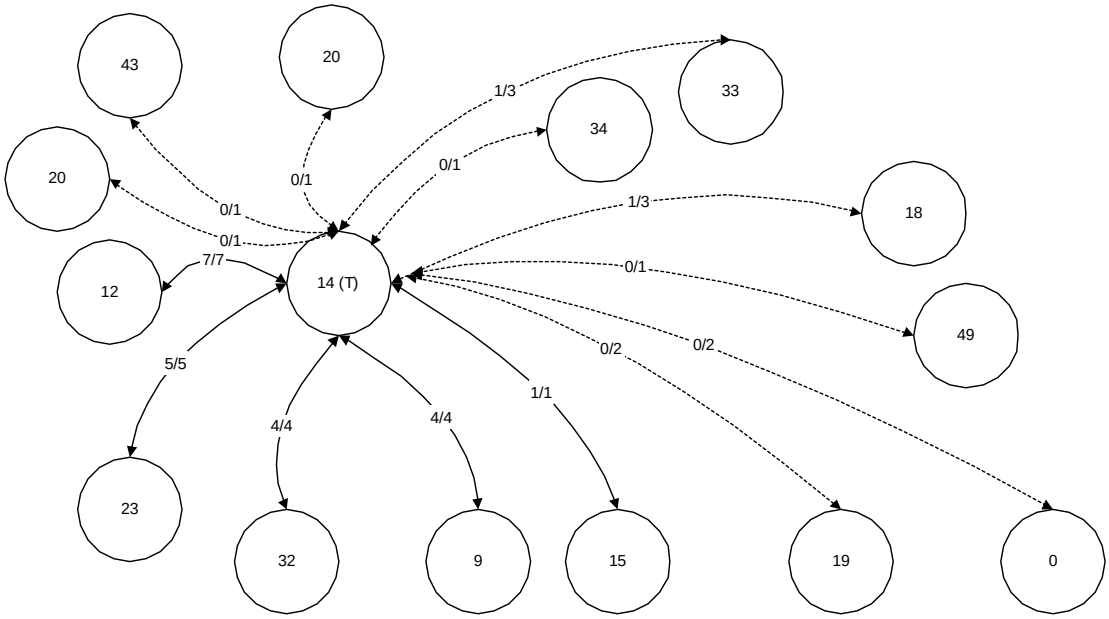

Figure 33. The sociogram after seventh execution cycle - stable clique

After the seventh execution cycle, the clique, consisting of agents 9, 12, 14, 15, 23 and 32, has reached stability and no further fluctuations were observed. Stability was achieved relatively early. Similar results were obtained in further experiments. The final state of the social network after 100 execution cycles is presented in figure 34.

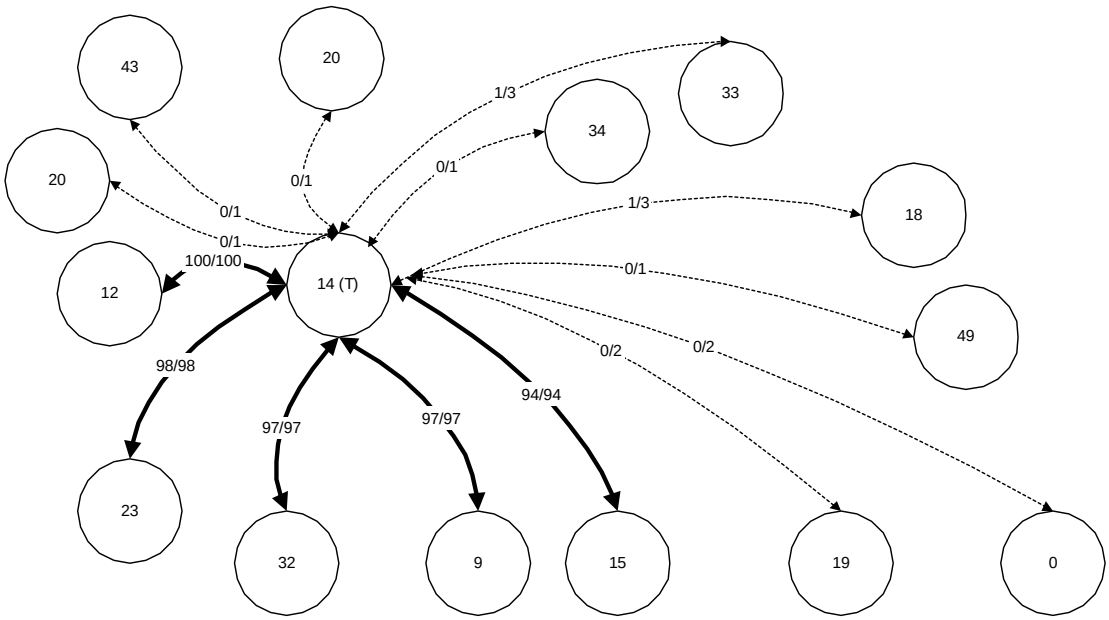

Figure 34. The final social network after 100 execution cycles

The investigation has confirmed that *cliques* do form in an artificial agent society, such as a society of agents in an abstract simulated environment. Firstly, the stability of a *clique* was investigated by observing the fluctuation of members over the period of execution. The *clique* has reached stability relatively early, after 6 execution cycles.

7.6 Summary

A partial implementation of the INDABA architecture, consisting of the upper two layers, was presented in this chapter. The agents in the abstract simulated environment were implemented using this reduced INDABA architecture. The social networks based approach was used as the main coordination mechanism and it was compared to an auctioning mechanism. A number of experiments were conducted that investigated all the parameters of the social networks based approach. Characteristics of the social network based approach were discussed, namely learning capability, specialisation of agents and the formation of *cliques*.

The next chapter presents a more comprehensive INDABA architecture implementation, again utilising the social networks based approach as a coordination mechanism.

Chapter 8: Experiments in the Simulated Robot Environment

The simulation and experiments described in chapter 7 were done with a high level of abstraction and a more realistic simulator was developed for the purpose of this thesis. This chapter presents a description of the simulator and the experiments conducted using the simulator. Section 8.1 summarises the main purpose of the simulator, while section 8.2 overviews the main components of the simulator. The simulation set-up, including a description of the robot population and environments used is given in section 8.3. Section 8.4 presents the experimental results.

8.1 Introduction

For the purpose of further experiments into the applicability of social networks as coordination tools in multi-robot teams, robot simulator software was developed. The main purpose of the simulator is to visualise the behaviour and movement of multi-robot teams during task execution. The robot simulator software itself is not a truly realistic simulation of the real-world, since many simplifications were made. A realistic robot simulation was considered at a certain stage (i.e. Webots [184]), but the limitations on social interaction models in readily availably robot simulations have necessitated the development of a simulator that focuses on multi-robot teams, albeit in a less realistic simulated environment.

8.2 Robot Simulator Overview

The robot simulator was developed in C++ in a Windows ™ environment. The robot simulator consists of the following components:

- The robot definition component, which encapsulates robots' behaviours.
- The society component, which maintains social links between robots.
- The environment component, which provides interaction with the simulated environment.
- The display component, which visualises task executions.

Each of the components will be described in more detail in the following sections. The main loop of the robot simulator is illustrated in algorithm 10.

```
For all Robots in team
   If Task not accomplished
      SetSensorReadings(Robot, Environment)
      Execute(Robot)
      ValidateAction(Robot, Environment)
      Display(Robot)
   EndIf
EndFor
```

Algorithm 10. The main loop of robot simulator

8.2.1 Robot Definitions Component

In order to keep the realism level high, robot-related functionality is kept and implemented independently in the robot definitions component. This facilitates a potential migration from the simulated robot to a real robot platform. The simulated robot architecture is based on a simplified INDABA agent architecture (refer to chapter 5), consisting of two layers:

- A controller layer that implements GoTo, AvoidObstacle, DetectObject and GrabObject behaviours.
- A combined sequencer and deliberator layer.

Such simplifications are justified by the fact that the task was one of the benchmark tasks for robotic teams: a simple foraging task. It was deemed to be unnecessary to implement a full inference engine based deliberator layer. Similarly, the interaction between the agents is handled by the society component of the robot simulator. Implementation of the interaction layer was therefore not deemed necessary. These simplifications should not impact on the "realism" factor in any manner.

The robot component receives input from the environment by reading sensor inputs from the environment. Based on the sensor inputs and the progress of task execution,

the combined sequencer and deliberator layer enables or disables behaviours in the controller layer. The desired robot actions are then sent to the environment component that evaluates their validity.

8.2.2 Display Component

The display component is used to monitor the execution of the allocated task. Minimum effort was spent on this component as it is of little relevance to the outcome of the experiments. A typical screenshot is presented in figure 35. White rectangles represent obstacles, small black rectangles represent "food", while the larger dark grey area represents "rough terrain". The symbol referred to by the white arrow on figure 35 indicates a robot.

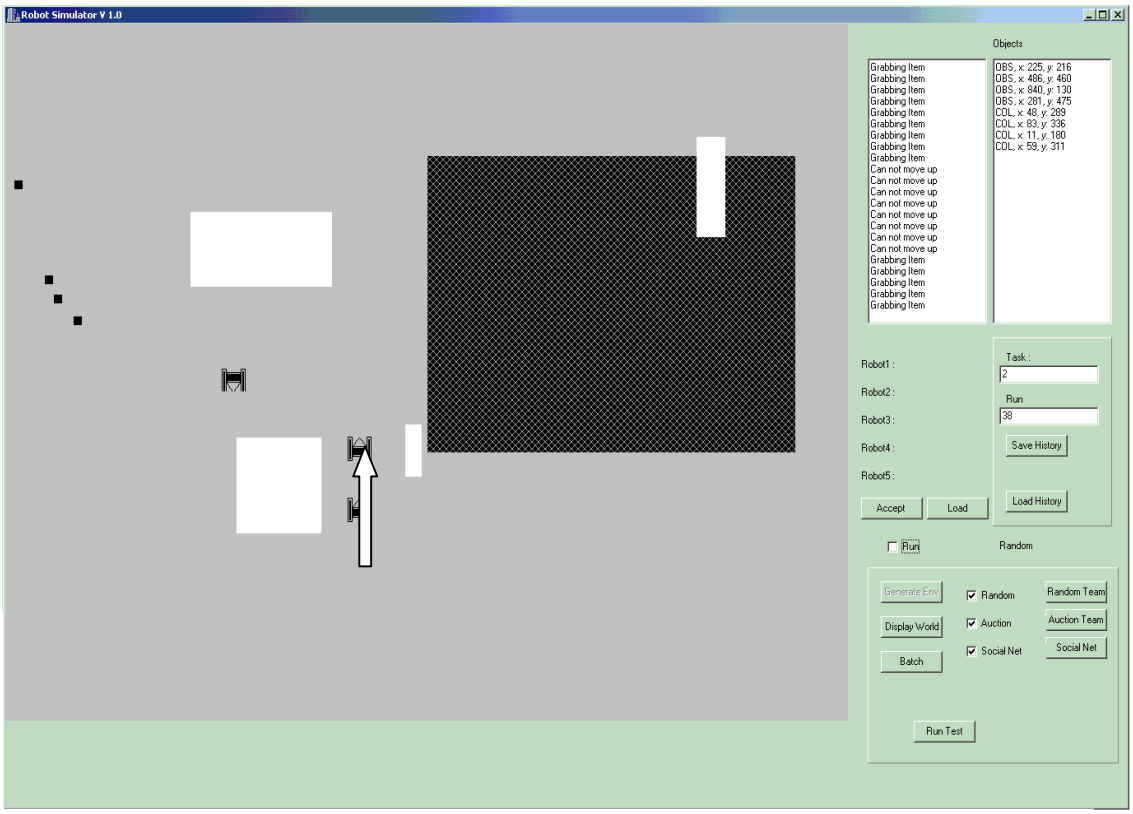

Figure 35. A Screenshot of Robot Simulator

8.2.3 Society Component

The main role of the society component is to facilitate team selection. In the INDABA architecture [157] the agent's interaction layer performs this task, as described in section 5.2. In the robot simulator, the society component allocates the tasks. This is done by finding the robot most suited to the task based on its performance history. If there is no performance history a robot is randomly selected. Then the strengths of social links are calculated using the definitions of kinship as given in section 7.1. The team allocation algorithm is given in pseudo-code in algorithm 11.

Announce Task T

For all Robots
If t(Robot, Robot, T)> t(leader, leader, T)
Leader = Robot
EndFor
If Leader found
For all Robot in society
Calculate s(leader, Robot, T)
EndFor
Select team based on highest s(leader, Robot, T)
Else
Select team based on auctioning
End Else

Execute Task

For all Robots in team
If Task successfully executed
Increase t (leader, Robot,T)
Else
Decrease t (leader, Robot,T)
EndFor

Algorithm 11. Task allocation and task success evaluation in simulated robot environment

8.2.4 Environment Component

The environment component simulates the real-world. It provides sensor readings based on a description of the environment, described in a set-up file which is loaded before execution begins. All the robot actions are validated within the limits and constraints of the environment, i.e. the obstacles and environment variables that influence the outcome of desired robot actions.

8.3 Simulation Set-up and Assumptions

For the purpose of this simulation, robot, environment and task attributes were selected to resemble potential real-world environments and problems. The setup is described with reference to the robots and environments used (refer to section 8.3.1) and the tasks to be performed (refer to section 8.3.2).

8.3.1 Robots and Environments

Each robot is defined by a predetermined set of attributes, as listed in table 22.

ROBOT ATTRIBUTE	POSSIBLE VALUES
LOAD	LOAD_SMALL
	LOAD_NORMAL
AVOIDANCE	AVOIDANCE_AVAILABLE
	AVOIDANCE_NOT_AVAILABLE
DRIVE	DRIVE_WHEEL
	DRIVE_TRACK
	DRIVE_LEG
SPEED	SPEED_LOW
	SPEED_MEDIUM
	SPEED_FAST
DETECTION RANGE	DETECTION_NORMAL
	DETECTION_LIGHT_ONLY
	DETECTION_ADVANCED
POWER	POWER_TETHERED
	POWER_SOLAR
	POWER_BATTERY

Table 22. Robot Attributes and possible values

It is important to note that the robots were defined by a set of attributes similar to that used in the experiments of chapter 7, but not identical.

The environments were defined in a similar manner, with environment attributes listed in table 23, where food refers to collectable items.

ENIVRONMENT ATTRIBUTE	POSSIBLE VALUES
TERRAIN	TERRAIN_NORMAL
	TERRAIN_ROUGH_AREA
LIGHT	LIGHT_NO_OBSTACLES
	LIGHT_SHADED_AREA
OBSTACLES	OBSTACLES_NONE
	OBSTACLES_FEW
	OBSTACLES_MANY
FOOD_DISTANCE	FOOD_FAR
	FOOD_CLOSE
FOOD_WEIGHT	FOOD_LIGHT
	FOOD_HEAVY
	FOOD_MIXED
FOOD_AVAILABILITY	FOOD_PLENTY
	FOOD_AVERAGE
	FOOD_SPARSE

Table 23. Environment attributes and possible values

Each robot action is expressed in terms of actuators, namely:

- Motor for left drive (on/off)
- Motor for right drive (on/off)
- Gripper motor (open/close)

The simulator does not prescribe a limit on the number of different robots and environments that could be created and used. However, for the purpose of experiments presented in this chapter, a total of 15 robots and 15 environments were created, all having randomly selected characteristics.

8.3.2 Tasks

While previous investigations focused on a simple foraging task using the same simulated robot environment [158], this thesis considers two tasks: scouting and then foraging.

For the purpose of these experiments, it is assumed that a robot is aware of its approximate position in relation to the environment. It is also assumed that a scouting robot has knowledge of the approximate position of the area where food is located. Although the robots have an idea of the approximate position of food, they do not have the precise position. Then, during the process of finding the best scout, if a robot reaches the approximate position, then that robot performs a spiral search to reach the food.

The following restrictions, similar but not identical to the rules given in section 7.1, are placed on the interaction between robots and the environment:

- If a robot's LOAD attribute is LOAD_SMALL, it cannot load food that has FOOD_WEIGHT attribute FOOD_HEAVY.
- If a robot's POWER attribute is POWER_SOLAR, it cannot move in an environment area that is in the shade.
- If a robot's POWER attribute is POWER_TETHERED, it is limited in range.
- The detection range is reduced in the shaded area and if a robot's DETECTION_RANGE attribute is DETECTION_LIGHT_ONLY, the robot cannot detect objects in shaded areas.
- A robot's progress is determined based on terrain, drive and speed (refer to section 7.1).

For each environment, the number of obstacles (depending on the value of the OBSTACLES attribute namely none, many (3-7 obstacles) or few (1-3 obstacles)) is randomly created. The same applies to food (depending on the value of the FOOD_AVAILABILITY attribute). Obstacles positioning is random, while food positioning is also random, but within the limits of the value of environment

parameter FOOD_DISTANCE. It is important to note that, due to the randomness of the environment attributes, in some cases a task could not be successfully executed at all. To illustrate, during the experiments conducted here, two of the environments were not successful due to the food being blocked by obstacles, so foraging was not possible. A consideration was given to recreating those two environment, but it was decided to use them unchanged. The reason for such decision is that, in real-world, it may happen that certain environments are just too difficult for robots to execute an allocated task [131].

For each environment, three different strategies have been used to select teams: (1) random team selection, (2) team selection based on auctioning and (3) team selection based on social networks. Using each strategy, two teams were formed, a scouting team and a foraging team. The scouting team consisted of only one robot, while three robots were in each foraging team. Each team was allowed to perform the task for a limited period of time (50 seconds).

If a robot did not complete the allocated task in the prescribed period of time (i.e. the robot did not find the food in the case of the scouting task, or did not return with the collected food to the home area in the case of the foraging task), it is considered that the robot failed to complete the task.

8.4 Simulation Results

In order to simulate a condition of market failure due to uncertainty, uncertainty about the task was introduced. Ucertainty was included by having incomplete information about environment attributes. For the purpose of this section, information was available for only FOOD_DISTANCE, FOOD_WEIGHT and TERRAIN attributes.

As mentioned in the previous section, each team had a limited size. If more than the allowed number of team members satisfied the auctioning bid, teams were selected randomly from these robots until the team size constraint has been met. For the purposes of this chapter, two experiments have been conducted.

For the first experiment, performance was evaluated when the same selection method is used for both the scouting team and foraging team. Performance was evaluated for the three different selection strategies (i.e. auction, social networks and random).

For the second experiment, the selection strategy used for the two teams were not necessarily the same. The strategy for selecting the scout was random selection. For the second experiment, three different studies have been done, where each study uses a different selection strategy for the foraging team.

Each experiment consisted of six simulations, where each simulation was run for forty-five execution cycles. Three execution cycles (each using a different selection strategy) were applied to each of fifteen randomly generated environments. It is important to note that each execution cycle builds on the social network, consisting of trust and kinship relationship, established during the previous execution cycle.

8.4.1 Results Using Same Selection Method

For the results reported in this section, the scout and foraging team selection utilise the same selection method. The results are illustrated in figure 36.

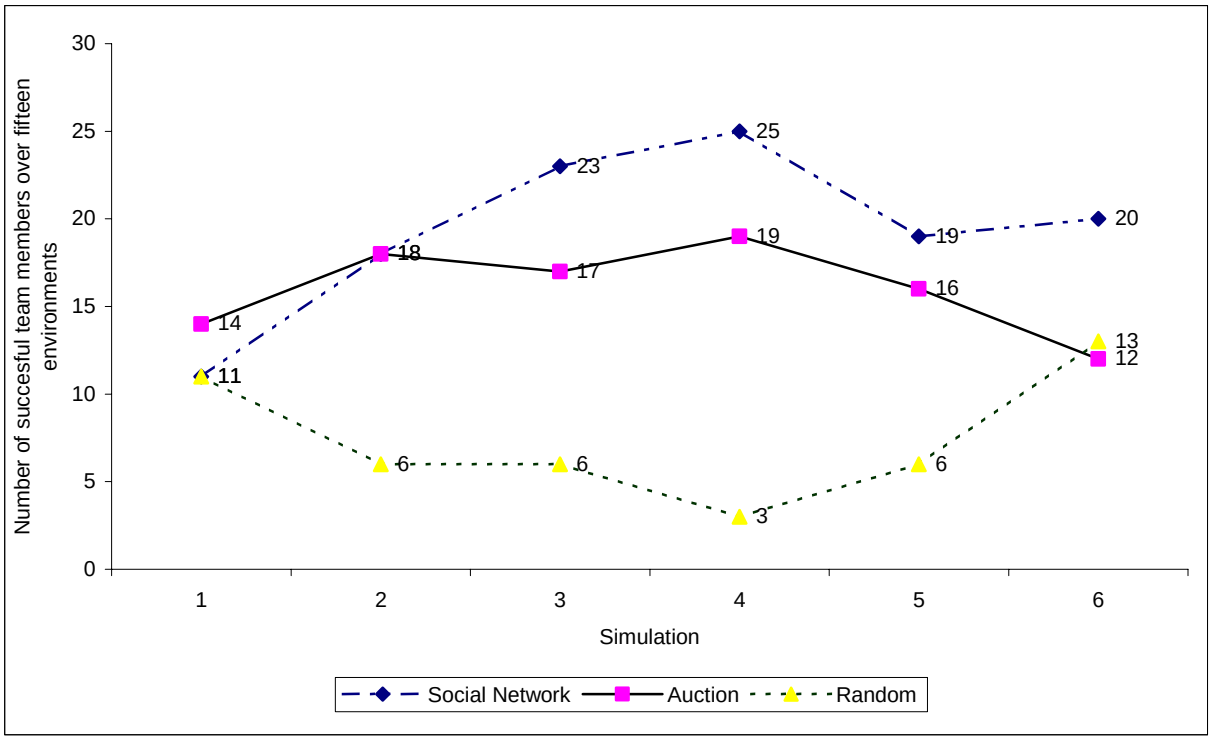

Figure 36. Comparative results of three selection methods over six simulations (same selection methods for both tasks)

It is important to note that for each of the three selection strategies, maximum number of successful agents per simulation is forty-five (three robots per team over fifteen randomly created environments). The social networks based team allocation mechanism on average performed better than the auctioning or random selection team allocation mechanisms.

In the first simulation, at which point there is no history, the social networks based team selection utilises auctioning. Therefor, there is no significant difference between the performances of the auctioning and social network team selection methods. Only after the third run, does the social networks approach start to outperform the auctioning method. The reason for this improvement in performance is due to the trust that was developed between the agents during the first two runs. In the third run, trust and kinship start to play the primary role in team selection. The performance of the auctioning system fluctuates, but on average does not improve. The random selection method performs far worse than the auctioning and social networks methods.

Fluctuations were observed in the performance of all three selection methods. It is important to note that these fluctuations are due to the uncertainty of the robot and food positioning, which are randomly selected for each environment (in case of food positioning) and for each task execution (in case of robot positioning). To quantify the uncertainty, in the experiment reported in this section robots were randomly positioned 2700 times (four robots (one scouting and three foraging robots) using three selection strategies for each of the fifteen environments in six simulations).

8.4.2 Random Scout Selection Method Simulation Results

This section reports results of using different selection method for the scout and foraging teams. Scout selection uses the random selection method, while the foraging team selection respectively utilise auctioning, social networks or random selection methods. The results are illustrated in figure 37.

Again the social networks based team allocation mechanism performed on average better than the auctioning and random selection team allocation mechanisms, but not

as well as for the previous experiments (refer to section 8.4.1). The lower performance can be attributed to the fact that whole task fails if the scout does not successfully complete its task. Random selection method does not guarantee that the best robot is selected for scouting task, while social networks based selection does (based on historical performance). The same selection method is therefore the recommended approach to fully utilise the advantages of the social networks based approach.

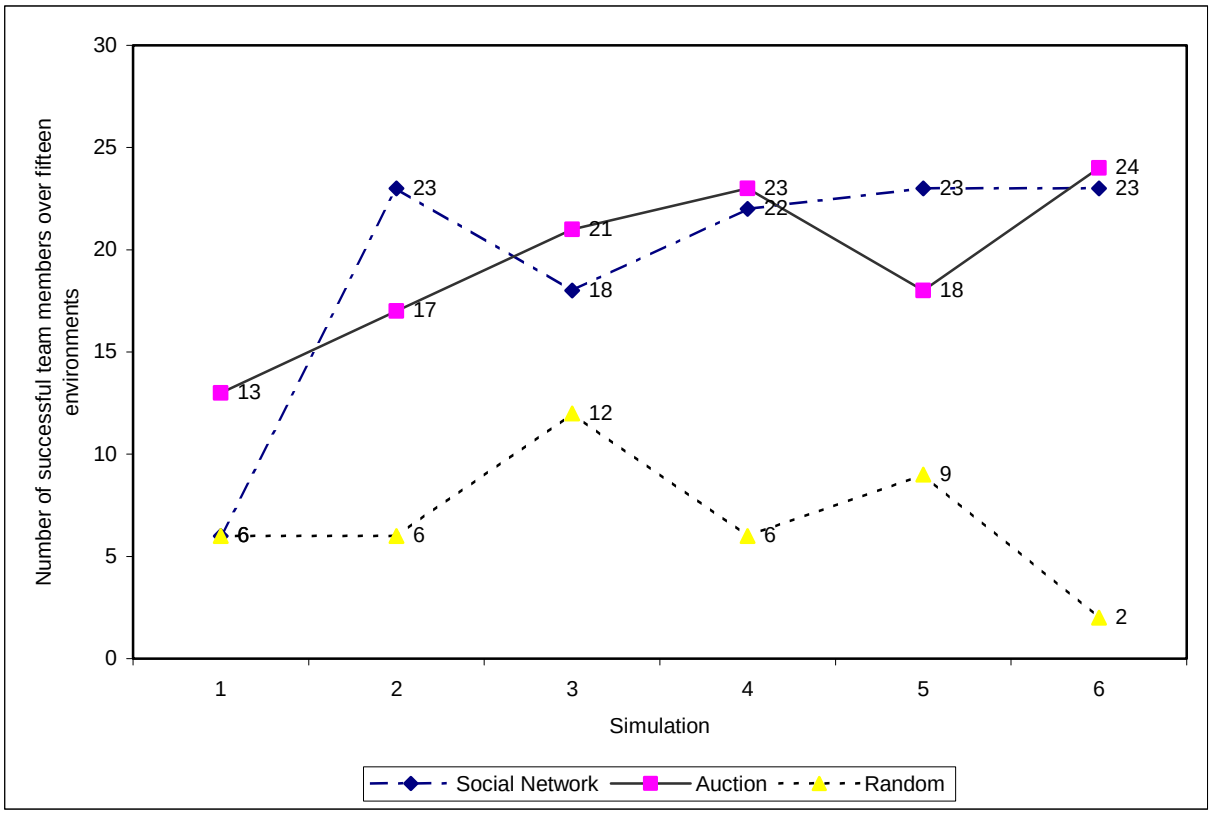

Figure 37. Comparative results of three selection methods over six execution cycles (inconsistent selection)

For both experiments it was noted that the social networks approach improved its performance over time, allowing for minor fluctuations. Social networks store information about relationships (kinship and trust) between members of the society. The stored trust relationship information is derived from the historical performance of the team members (refer to section 8.2.3 and algorithm 11). This information is used in the process of team selection, and leads to improvement in performance of the selected team.

The improvement in performance, due to the better team selection method based on kinship and trust relationships, can be seen as a learning capability.

8.5 Summary

The INDABA architecture was used as the agent architecture in a simulated robot environment, which was presented in this chapter. Scouting and foraging tasks were simulated in this environment and the results were presented and discussed. The social networks based approach was used as the main team selection mechanism. A comparison was made to alternative team selection strategies, namely auctioning and random selection. The results of conducted experiments indicate that social networks based approach performs better than alternative team selection strategies.

The next chapter presents a full INDABA implementation in a physical environment, using a readily available robotic platform.

Chapter 9: Experiments in a Physical Environment

The new agent architecture, INDABA, was proposed in chapter 5. INDABA was partially implemented for the purpose of the simulations and experiments described in chapters 7 and 8. The agents based on INDABA were implemented and they executed allocated tasks. The fourth layer of INDABA, the interaction layer has proved to be a valuable addition to the standard three layers. The interaction layer allowed for ease of implementation of various team selection methods. As a final confirmation of the applicability of INDABA to a robot architecture, this chapter provides results from a physical environment implementation. An introduction to the chapter is given in section 9.1. Section 9.2 describes the physical environment set-up, including an overview of the physical robots used, and environment types. Section 9.3 describes an implementation of the INDABA architecture to a hardware platform with limited capability, namely LEGO Mindstorms [185]. The social networks approach was used for robot selection for a scouting task. The results are presented in section 9.4.

9.1 Introduction

Section 3.2 discussed Shakey, a robotic architecture [137] implemented in physical environment. One of the lessons learned from the Shakey project was that although a certain architecture may perform well in theory, the same architecture, when implemented in a physical environment, may not perform up to the expectations [120]. The view proposed by some of the leading researchers [28][112] and adopted in this thesis is that a robotic architecture must be implemented in a physical environment in order to accurately evaluate its performance. To prove the applicability of the INDABA architecture, and more specifically the social based approach to task allocation, the implementation was done in a physical environment. However, the focus of this thesis is not in creating a sophisticated multi-robot team environment. Instead, a simplistic physical environment was chosen for the purpose of this thesis, to show proof of concept.

9.2 Physical Environment Set-up

The physical environment used in this thesis uses standard "off-the-shelf" hardware and many simplifications (in comparison with a real-world application) were made. For example, the navigation method used is dead-reckoning (as applied to a LEGO platform [64]). The dead-reckoning navigation method is not ideal, especially considering the selected platform (due to the inaccuracies of the available sensors).

The physical environment set-up can be described by describing its three main components: robotic platform, robot population and the environment set-up.

9.2.1 Robotic Platform

For the purpose of a physical robot implementation, LEGO Mindstorms robotic platform was selected. LEGO Mindstorms robotic platform [185] was developed by LEGO and it was inspired by research done on MIT's Programmable Brick [162]. LEGO Mindstorms is an easy to use, reliable and cheap robotic platform that comes with a variety of development tools, the majority of which were developed by LEGO and the LEGO users community.

At the heart of every LEGO robot is a RCX Brick [185], a simple computer that supports the concurrent execution of up to 10 processes at a time. The number of available variables is 32 for global variables and 16 for local variables. All variables are of 16-bit signed integer type. An RCX has a Hitachi H8/300 CPU and 32K RAM, and it also has a limited communication capability. It is equipped with an IR port that is capable of sending and receiving messages. The LEGO Mindstorms was investigated in [155].

There are a few major shortcomings of the LEGO Mindstorms communication capability, as implemented in standard RCX, that makes the communication unreliable and of limited use:

- The communication is done via an IR port and as such it is basically line of sight communication with a limited range (in ideal conditions, about 10 meters).

- The messaging protocol is extremely simple and there is no addressing mechanism. This in turn means that it is impossible to send a message to a particular robot in a multi-robot team.

- The application programming interfaces (APIs) for LEGO IR tower are usually a third-party software with limited usability and they are usually not error-free. Often the provided APIs are not general enough to provide simple integration into a more complex software application.

The RCX Brick also supports numerous standard sensors (up to three at a time) and it can control up to three actuators. The supported sensors are very basic and not very accurate. The standard sensors include light, temperature, touch and rotation sensors. The standard actuators include micro motors and light sources.

The RCX can be programmed using a variety of programming languages. The two most popular LEGO programming languages are:

- Not Quite C (NQC) [182], which is a subset of the C programming language, adapted for RCX, and
- LASM (Lego ASEMBLER) [185].

NQC [182] was selected as the programming language for robot implementation.

9.2.2 Robot Population

Unfortunately, the choice of agent attributes was mainly limited by the availability of LEGO Mindstorms sensors and actuators, which tend to be very basic (refer to section 9.2.1). In order to increase the variety of attributes, some of the agent attributes are

implemented as software features and they depend on particular robot programming. The list of attributes, together with possible attribute values, is given in table 24.

AGENT ATTRIBUTE	POSSIBLE VALUES
OBSATCLE AVOIDANCE	NO_AVOIDANCE
	AVOIDANCE
DRIVE	DRIVE_WHEEL
	DRIVE_TRACK
SPEED	SPEED_MEDIUM
	SPEED_HIGH
DETECTION	NORMAL
	LIGHT_ONLY
POWER	TETHERED
	BATTERY

Table 24. Robot attributes and possible values

Due to the limited availability of LEGO Mindstorms kits at the University of Pretoria, the robot population was restricted to six robots. The attribute values of agents's attributes are given in table 25. Each row represents one of the agents in the population.

ID (TYPE)	OBSTACLE AVOIDANCE	DRIVE	SPEED	DETECTION	POWER
0	AVOIDANCE	DRIVE_WHEEL	SPEED_MEDIUM	LIGHT_ONLY	BATTERY
1	NO_AVOIDANCE	DRIVE_WHEEL	SPEED_HIGH	LIGHT_ONLY	TETHERED
2	AVOIDANCE	DRIVE_WHEEL	SPEED_MEDIUM	NORMAL	BATTERY
3	NO_AVOIDANCE	DRIVE_WHEEL	SPEED_HIGH	NORMAL	TETHERED
4	AVOIDANCE	DRIVE_TRACK	SPEED_SLOW	NORMAL	BATTERY
5	AVOIDANCE	DRIVE_TRACK	SPEED_SLOW	LIGHT_ONLY	BATTERY

Table 25. Robot population

An example of a robot (type 5) defined by the 5-tuple (AVOIDANCE, DRIVE_TRACK. SPEED_MEDIUM, LIGHT_ONLY, BATTERY) is given in figure 38.

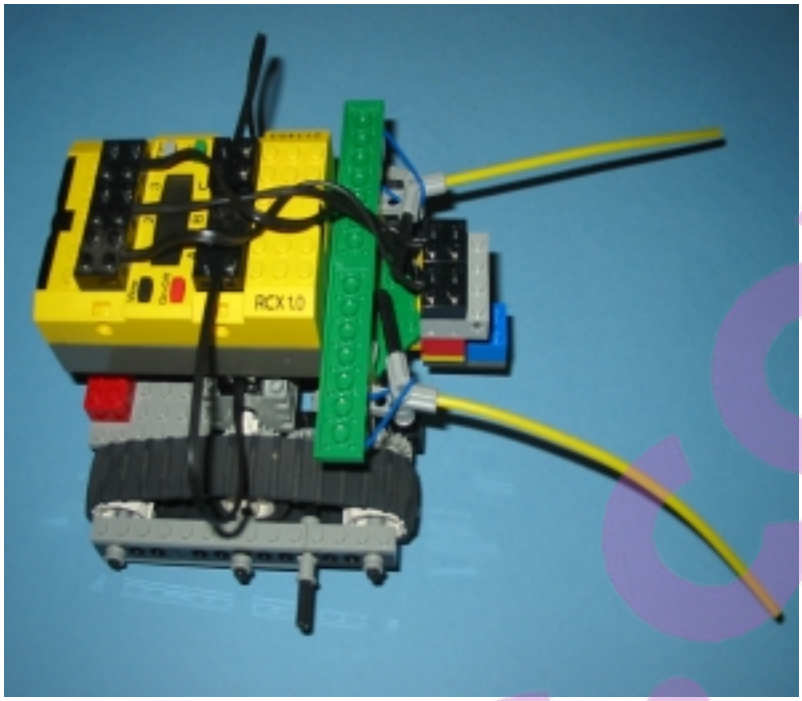

Figure 38. An example of a robot used in the experiments (type 5).

Robot type 5 utilises differential steering and it uses tracks as the main means of propulsion. As obstacle detection mechanism, two antennae are used. The robot has a simple light sensor for detection of collectable objects.

An example of another robot (type 3), defined by the 5-tuple (AVOIDANCE, DRIVE_WHEEL, SPEED_MEDIUM, NORMAL, BATTERY) is given in figure 39.

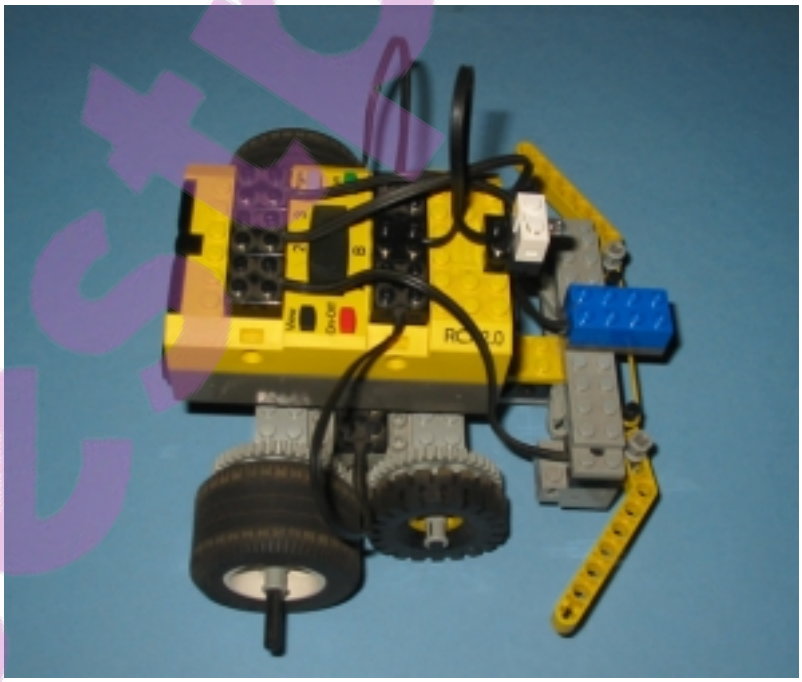

Figure 39. An example of a robot used in the experiments (type 3).

Robot type 3 again utilises differential steering but uses wheels as the main means of propulsion. Two bumpers are used as obstacle detection mechanism sensors. The robot has a simple light sensor for detection of collectable objects. Furthermore, the robot type has a Light Emitting Diode (LED) that allows for object detection in dark environments.

9.2.3 Environment Set-up and Types of Environment

For the purpose of the experiments in a physical environment, various environments were created. Each environment can be described by a set of attributes. The environment attributes and valid attribute values are given in table 26.

ENIVRONMENT ATTRIBUTE	POSSIBLE VALUES
TERRAIN	NORMAL
	ROUGH_AREA
LIGHT	NO_SHADED_AREA
	SHADED_AREA
FOOD DISTANCE	FAR
	CLOSE
OBSTACLES	NO_OBSTACLES
	OBSTACLES

Table 26. Environment attributes and possible values

For the purpose of this thesis, four environments were created. These environments are described next. The first environment is described by the 4-tuple (ROUGH_AREA, NO_SHADED_AREA, FAR, NO_OBSTACLES). The environment (illustrated in figure 40) has two areas of a rough terrain.

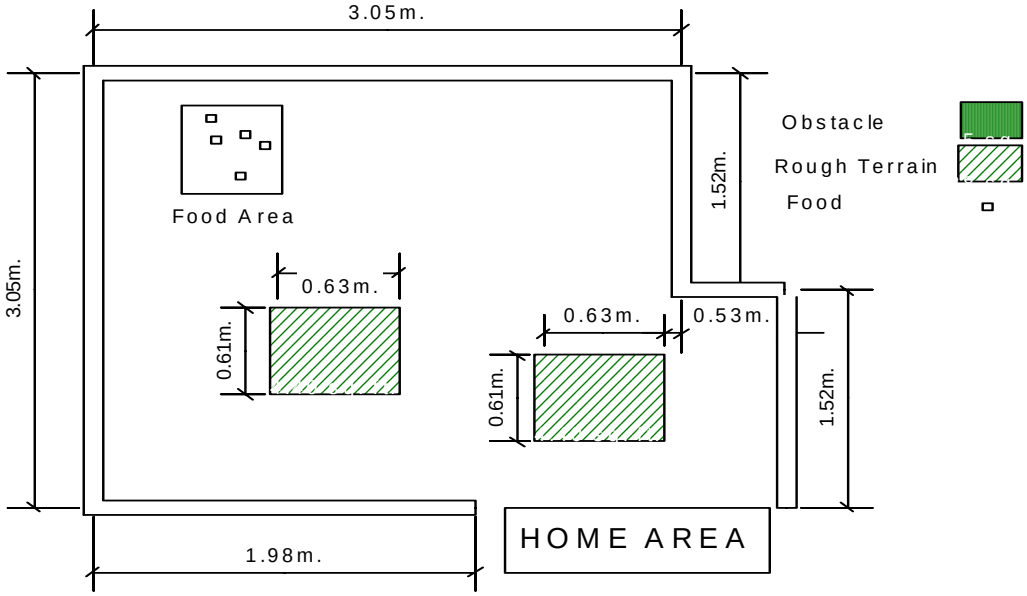

Figure 40. First environment used in experiments

The second environment used in these experiments is described by the 4-tuple (ROUGH_AREA, NO_SHADED_AREA, FAR, OBSTACLES) and is illustrated in figure 41.

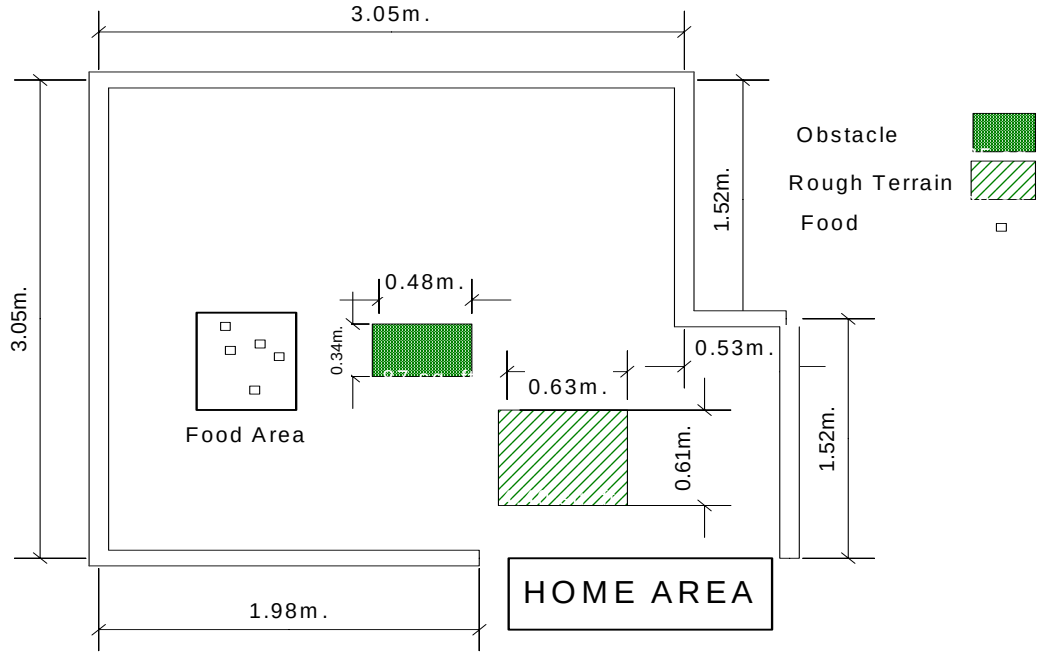

Figure 41. Second environment used in experiments

The 4-tuple (ROUGH_AREA, NO_SHADED_AREA, CLOSE, NO_OBSTACLES) describes the third environment, as illustrated in figure 42.

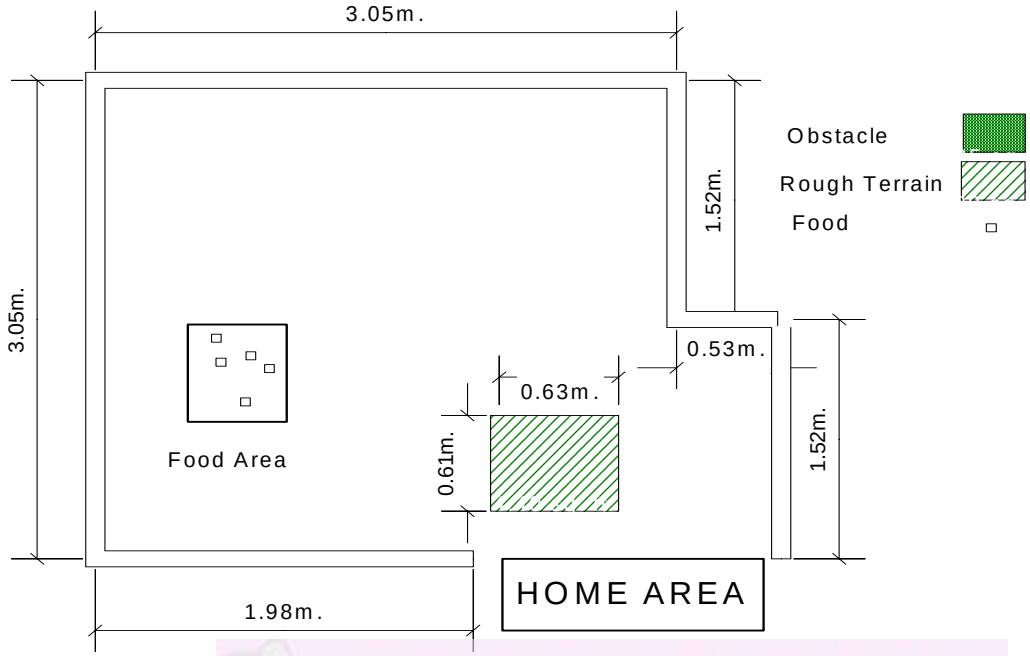

Figure 42. Third environment used in experiments

The fourth and last environment is described by the 4-tuple (NORMAL, SHADED_AREA, FAR, OBSTACLES) and is illustrated in figure 43.

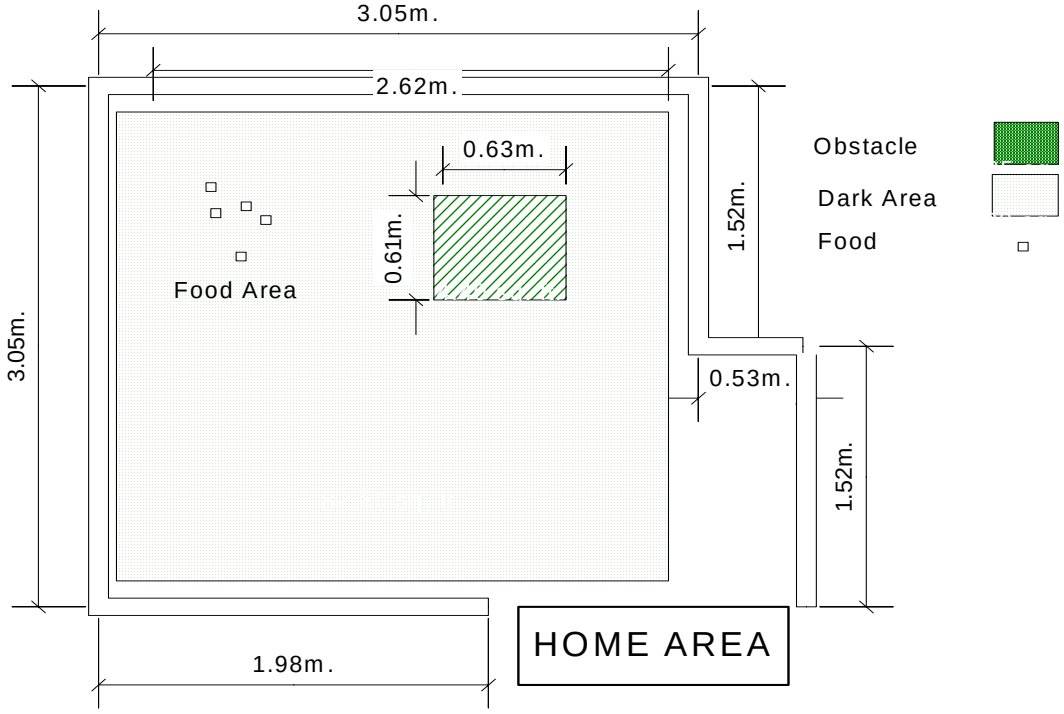

Figure 43. The fourth environment used in experiments.

Table 27 summarises the environments used in the experiments, together with their attribute values.

ID (TYPE)	TERRAIN	LIGHT	FOOD DISTANCE	OBSTACLES
0	ROUGH_AREA	NO_SHADED_AREA	FAR	NO_OBSTACLES
1	ROUGH_AREA	NO_SHADED_AREA	FAR	OBSTACLES
2	ROUGH_AREA	NO_SHADED_AREA	CLOSE	NO_OBSTACLES
3	NORMAL	SHADED_AREA	FAR	OBSTACLES

Table 27. Summary of environment types used in experiments

9.3 INDABA Implementation

The INDABA implementation presented in this chapter splits the four layers of INDABA into two groups, one implemented in the robots and the other one implemented as an application on a desktop PC equipped with an IR tower for communications with the robots. The reason for such implementation is that the

computational requirements of a full INDABA four-layer architecture exceeded the computing capabilities of the selected robotic platform.

Communication between the deliberator layer and the sequencer layer uses the infrared channel. The Phantom Application Programming Interface (API) [183] is used for communication purposes.

The Phantom Control API allows for direct access to variables stored on the RCX Brick. This mechanism was utilized to set active goals (by setting the B variable) and for retrieving the status of each goal (by getting the G variable). In order to reduce communication traffic (as the RCX can send and receive only a byte at time) one variable was used for setting active behaviours and another for retrieving the status of each behaviour. Behaviours and statuses were encoded using simple binary encoding. The overall architecture is illustrated in figure 44.

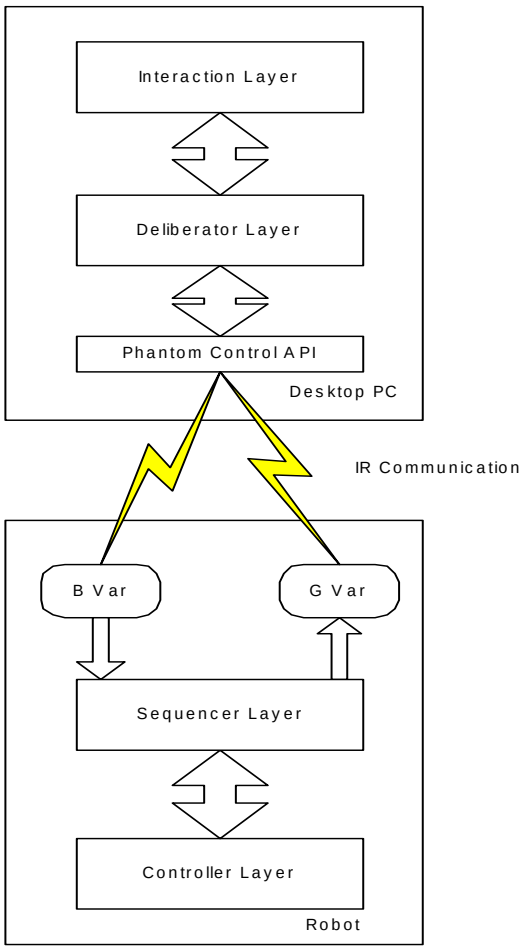

Figure 44. Implemented Hybrid Architecture

Due to the fact that the implemented method of communication allows only a PC to initiate communications, polling is used to gather date from robots. Polling occurs every 250ms.

9.3.1 Implemented Robot Components

Because of the processing power limitations of LEGO Mindstorms, only the lower two layers of INDABA could be implemented in the selected robotic platform. The lower two layers, that are computationally less demanding, are the controller layer (together with corresponding behaviours that are implemented therein) and sequencer layer.

9.3.1.1 The Controller Layer

The controller layer consists of several behaviours, namely basic behaviours and synthesised behaviours. Synthesised behaviours are combinations of basic behaviours.

The basic behaviours, provided by the NQC implementation [182], are:

- *On* (OUTPUT); this simple behaviour activates output OUTPUT indefinitely.
- *OnFor* (OUTPUT, TIME); this behaviour activates output OUTPUT for a period TIME (TIME is expressed in tenths of a second).
- *Off* (OUTPUT); deactivates output OUTPUT.
- *OnRev* (OUTPUT); reverses the polarity of output OUTPUT. If a motor is connected to output OUTPUT, this behaviour will reverse its direction.

The synthesised behaviours are implemented as tasks in NQC terminology [182]. The synthesised behaviours include:

- *beh_move_for* (DIRECTION, DURATION). This behaviour encapsulates the details in how motion is achieved. Only direction of the movement and its duration are provided as input to this behaviour.

- *beh_detect.* This behaviour constantly monitors the input received from the light sensor. If the input exceeds a certain, predefined value, the behaviour terminates, signalling the detection of a food object.
- *beh_spiral_search.* A behaviour that executes increasing spiral search.
- *beh_safe_move.* A behaviour that allows for a movement with obstacle detection. If an obstacle is detected, an attempt is made to avoid the obstacle.
- *beh_send.* The behaviour that transmits the coordinate where food is detected. It is important to note that the coordinates are calculated using a dead-reckoning navigation approach and as such is susceptible to error.

The synthesised behaviours were sufficient for a simple scouting task. These behaviours are activated and deactivated by the next layer of INDABA, the sequencer layer.

9.3.1.2 The Sequencer Layer

The sequencer layer combines the behaviours, as implemented in controller layer, in order to achieve certain goals. The sequencer is also implemented as a task in NQC terminology. It uses NQC commands *start task* and *stop task* to activate and deactivate the behaviours in the controller layer. For the purpose of the scouting tasks, three goals are defined:

- *GOAL_AREA.* This goal is achieved when a robot is in the target area that is the start of the approximated food area.
- *GOAL_FIND.* This goal is activated when a robot is in the target area. It is achieved when a behaviour *beh_detect* terminates, indicating the detection of a food object.
- *GOAL_SEND.* This goal transmits the coordinates of the area where food was detected.

The goals are a combination of behaviours. For the purpose of this application the combinations are hard-coded. Table 28 illustrates the implementation of the sequencer layer.

GOAL	ACTIVE BEHAVIOURS				COMPLETION CRITERIA
	beh_safe_move	*beh_detect*	*beh_spiral_search*	*beh_send*	
GOAL_AREA	ACTIVE	ACTIVE			Time or *beh_detect*
GOAL_FIND		ACTIVE	ACTIVE		*beh_detect*
GOAL_SEND				ACTIVE	*beh_send*

Table 28. Illustration of the implemented sequencer layer.

The sequencer layer constantly monitors a change in value of the variable G (see section 9.3). If a change is detected, the variable is decoded and the appropriate goal is activated, which in turn activates the corresponding behaviours (as in table 21).

9.3.2 Components Implemented in the Desktop PC

The higher two layers, i.e. the deliberator and interaction layers of INDABA are implemented in a desktop PC Windows ™ environment. The programming language used was C++. The implementation of deliberator and interaction layers is very similar to the implementation as described in chapter 7, and there was extensive re-use of the code.

9.3.2.1 The Deliberator Layer

For the purpose of this particular INDABA implementation, a simple backward and forward chaining inference engine was developed. More on backward and forward chaining can be found in many AI textbooks, such as [170]. It is important to note that the backward chaining process is modified to suit the execution in a robot environment, as follows.

The deliberator layer loads the rules from a text file. When a goal is selected, the rules are back-chained until the first sub-goal is identified. The first sub-goal is a goal that cannot be back-chained further. The sub-goal is then passed as a goal to the sequencer layer. When the sequencer layer returns the sub-goal results, the inference engine

determines the next sub-goal and the process continues until the value of the goal can be determined.

To illustrate the process, consider the (very simple) set of rules as implemented for the purpose of the experiments described in this chapter. The implemented rules are:

Rule 1 : IF *GOAL_AREA* THEN *GOAL_FIND*
Rule 2: IF *GOAL_FIND* THEN *GOAL_SEND*
Rule 3: IF *GOAL_SEND* THEN *GOAL_SCOUT*

Consider the activation of goal *GOAL_SCOUT*. The inference engine back-chain rules until it gets to the first sub-goal from which it cannot back-chain further. That sub-goal is *GOAL_AREA*. The *GOAL_AREA* is then sent as a goal to the sequencer layer, utilising the mechanism described in section 9.3. Once the goal is achieved, the whole process repeats, but this time the first sub-goal is the *GOAL_FIND* and the process continues until the *GOAL_SCOUT* is satisfied or until the allocated time is exceeded.

The activation of a goal and the allocation of time to the task execution are done by the fourth and last layer of INDABA, the interaction layer.

9.3.2.2 The Interaction Layer

The interaction layer consists of two main components: the task allocation and task evaluation components. The task allocation component utilises the social networks based approach, as described in section 7.1.

- *Task Allocation*

The algorithm starts with the task details propagation. For the purpose of the experiments and the selected task (scouting) implemented in this chapter, the task details consisted only of a time constraint, defined as maximum execution time.

Each of the available robots evaluates its own suitability to the task, by examining its own historical performance (using a trust relationship to itself, as described in section 7.1). The algorithm is similar to the general team leader selection algorithm 7.

The robot with the highest score is selected. In the case where the highest scoring robot is not available for the task, the next best one (based on kinship relationship as described in 7.1) is selected for the task execution. Setting the goal GOAL_SCOUT in its deliberator layer then activates the selected robot.

- *Task Evaluation*

If a robot is still executing when the allocated time expires, (the value of the goal *GOAL_SCOUT* in the deliberator layer is unknown), the agent is considered unsuccessful in the task.

If the execution of the task was successful (the value of the goal *GOAL_SCOUT* is true), its affinity to the task is increased. In other words, its own trust rating relative to a particular task's details improves (this represents its own historical performance; refer to section 7.3.). This in turn determines an agent's affinity to a particular task type.

9.4 Results

In order to provide some historical data, robots were initially randomly selected for the scouting task. For each environment, a robot was ten times randomly selected from the population of six robots (refer to table 25) and tasked with the scouting task.

To prove the validity of the social networks based team allocation mechanism, the fifth environment was randomly created and the social networks based team allocation mechanism was compared to random selection.

The results of random selection and a brief discussion on encountered issues for each environment are presented next, followed by the comparison of the social networks based approach with random selection.

9.4.1 Random Selection Results

The results of robots execution in various environments are presented in table 29.

ENVIRONMENT	SELECTED ROBOT	RESULT (Min:Sec)
1	4	0:23
1	5	FAIL (Lost Position)
1	4	0:42
1	2	FAIL (Rough Area)
1	0	FAIL (Rough Area)
1	1	FAIL (Range)
1	0	0:45
1	5	0:57
1	0	FAIL (Rough Area)
1	3	FAIL (Range)
2	5	1:05
2	2	FAIL (False Detect)
2	0	FAIL (Avoidance)
2	2	FAIL (Avoidance)
2	5	FAIL (Avoidance)
2	5	1:15
2	0	0:54
2	2	FAIL (Rough Area)
2	4	FAIL (Avoidance)
2	1	FAIL (Avoidance)
3	0	0:22
3	5	0:15
3	4	1:03
3	1	0:20
3	2	FAIL (Rough Area)
3	1	0:12
3	2	0:53
3	1	0:05
3	2	0:06
3	5	0:43
4	4	0:37
4	1	FAIL (Range)
4	1	FAIL (Range)
4	0	0:43
4	3	FAIL (Range)
4	1	FAIL (Range)
4	3	FAIL (Range)
4	5	0:16
4	4	0:48
4	2	FAIL (Rough Area)

Table 29. The results of random selection robot scout execution in physical environments.

The task was considered unsuccessful if a robot could not complete it in less than two minutes. The results reflect the time that it took a robot to complete the task, or alternatively the reason why it failed.

Numerous problems were encountered during the robots' execution in a physical environment, mainly related to randomness in the environment and physical robot attributes. The two most common problems are:

- *Changes in Lightning Conditions*

The LEGO Mindstorms light sensor is very sensitive to changes in light conditions. The robotic lab had a window and while every effort was made to restrict the light, the slightest change required recalibration. It was impossible to use the same settings in the morning and afternoon. To counter these effects, the robot was "trained" to recognise appropriate light sensor input for a food object every time before its execution. In other words, each robot was calibrated before execution.

- *Changes in Navigation Accuracy*

Inevitably, after a few execution cycles robots frequently lost their capability to move forward in a straight line, due to dust and residue build-up in their drive assembly. Affected robots then start veering to one side. This in turn leads to an imprecise spiral search process, and in extreme cases, the robots would get stuck turning in only one direction. Dead-reckoning navigation method then becomes useless. Furthermore, this inaccuracy also affected the obstacle avoidance algorithm, which was particularly visible when considering the results presented in table 26 for the environment 2. Although a simple obstacle avoidance mechanism was implemented, the sensors did not always accurately detect an obstacle and even when they did, the performed corrective action sometimes lead to potential loss of positioning.

Table 30 provides a summary of results per robot.

ROBOT	ATTEMPTED	SUCCESS	RATING (SUCCESS/ATTEMPTED)
0	7	4	0.57
1	8	3	0.38
2	8	2	0.40
3	3	0	0.00
4	6	5	0.83
5	8	6	0.75

Table 30. The results sorted by robots

9.4.2 Social Network Based Selection vs Random Selection

In order to check the validity of the social networks based approach, a comparison was made to a random selection approach. The social networks based approach was used for scouting team selection. As in the previous experiment, the scouting team was limited to one member. The trust relationship between agents was non-existent, however agents had trust in their own capabilities (based on historical performance, refer to table 29).

The kinship relationship table between the robots is pre-calculated and given in table 31 (the maximum strength is 1.0 and the minimum 0.0 – refer to formula 7.1).

	Robot 0	Robot 1	Robot 2	Robot 3	Robot 4	Robot 5
Robot 0	1.0	0.5	0.83	0.33	0.67	0.5
Robot 1	0.5	1.0	0.33	0.83	0.17	0.0
Robot 2	0.83	0.33	1.0	0.5	0.5	0.67
Robot 3	0.33	0.83	0.5	1.0	0.0	0.17
Robot 4	0.67	0.17	0.5	0.0	1.0	0.83
Robot 5	0.5	0.0	0.67	0.17	0.83	1.0

Table 31. Kinship between the robots

A sociogram to illustrate the kinship based social network is given in figure 45. For the purpose of this illustration, a strong kinship relationship is defined as a kinship with strength of 0.8 or more, and is illustrated by a thicker link between robots.

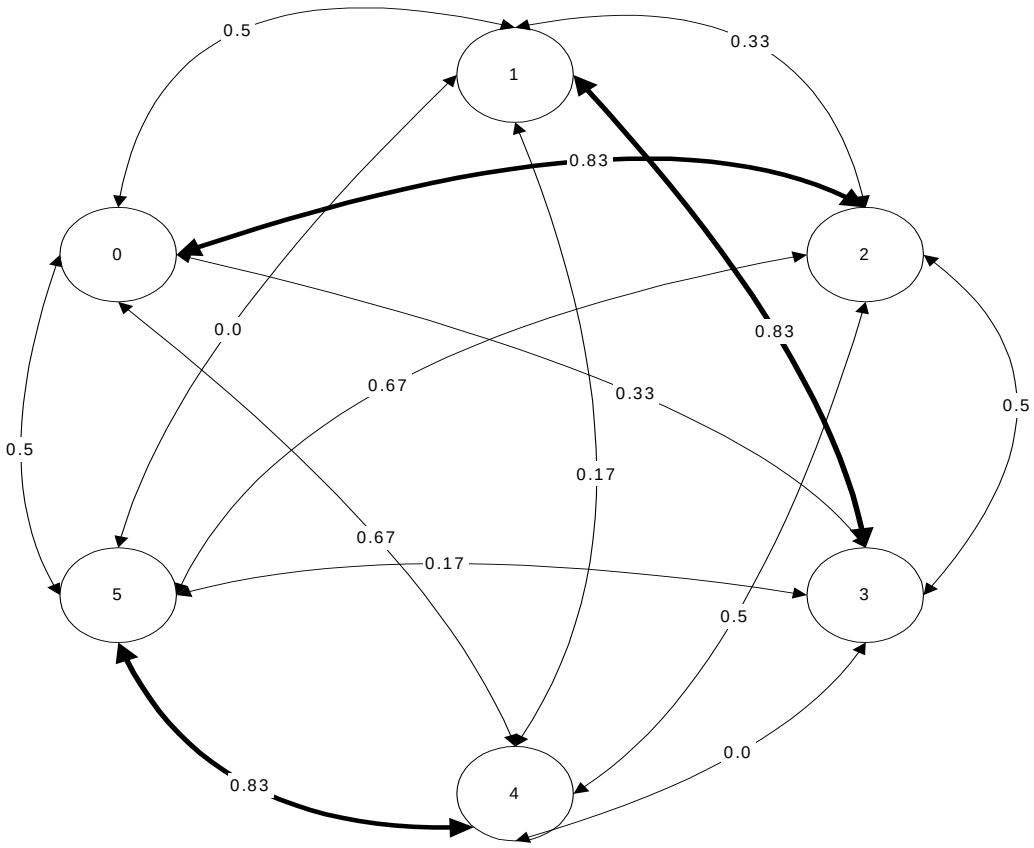

Figure 45. The sociogram of kinship relationship between robots

In order to introduce uncertainty, a new environment was created, as illustrated in figure 46.

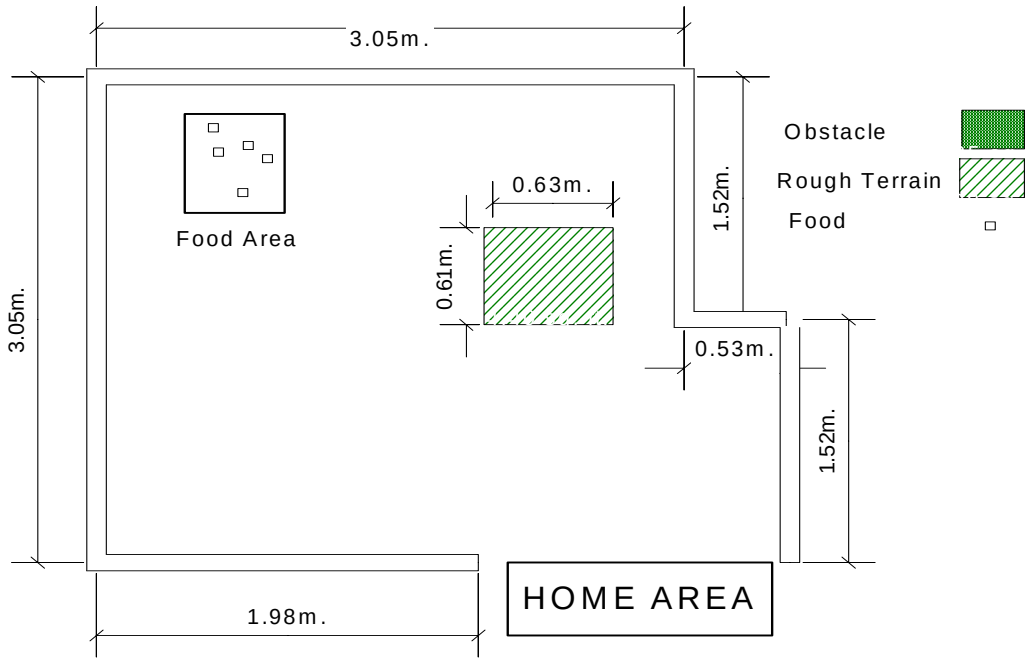

Figure 46. The fifth (test) environment used in experiments.

It is important to note that the test environment was a relatively easy one, as it had only one rough area and no shaded areas or obstacles. The social networks based approach was verified as follows: The best performing robot (robot 4) was considered unavailable. The social networks based approach therefore selected the next best available robot, based on the kinship relationship (refer to section 6.4.4). The result of the selected robot's execution was then compared to that of the randomly selected robot. For each scout selection method (i.e. social networks based and random), ten simulations have been done. Each simulation executed for a maximum of two minutes.

The task was considered unsuccessful if a robot could not complete it in less than two minutes. The results given in table 32 reflect the time that it took a robot to complete the task, or alternatively the reason why it failed. The results of the scouting task executions are presented in table 32.

SOCIAL NET SELECTED SCOUT	RESULT (Min:Sec)	RANDOMLY SELECTED ROBOT	RESULT (Min:Sec)
5	0:41	2	0:23
5	1:03	5	0:54
5	FAIL (False Detect)	1	FAIL (Range)
5	0:44	2	FAIL (Rough Area)
5	0:53	0	FAIL (Lost Position)
5	1:12	1	FAIL (Range)
5	0:57	0	0:26
5	0:48	5	0:39
5	0:38	0	FAIL (Rough Area)
5	FAIL (Lost Position)	3	FAIL (Range)

Table 32. Comaparisson of the results

The fluctuations in the time required for task execution were related to the intial robot positioning, changes in lightning conditions (which influenced light sensor readings) as well as the general failure of robots to maintain a straight direction without veering to one side.

The scout selected using the social networks based approach, successfully executed the task eight times, while randomly selected scouts succesfully executed the task only four times. The social networks based approach is therefore more reliable than random selection method, as demonstrated in this experiment.

9.5 Summary

This chapter presented a full INDABA agent architecture implementation, applied to physical robots. The primary purpose of this chapter was to verify the applicability of INDABA to physical robots. The secondary purpose of this chapter was to investigate limited application of the social networks selection method to physical robots. It is important to note that the social networks approach was the focus of chapter 7, where the social networks approach has been investigated in great detail in simulated environments. In this chapter, the social networks approach was implemented in a much-simplified manner.

The chosen task and chosen robotic platform were simplistic, as the focus was not on implementing a realistic real-world environment. Robots were given a scouting task to complete within a time constraint.

While the full physical implementation of a simulated environment was somehow restricted, it nevertheless provided proof of the applicability of the INDABA architecture to real-world robotic applications.

The social networks based approach, albeit using only a kinship relationship, performed better than a random selection strategy.

The next chapter summarises the work presented in this thesis.

Chapter 10: Conclusion

This chapter presents a brief summary of the contributions and findings of this thesis, as well as some discussion on future research. The contribution and findings related to the new proposed INDABA framework are discussed in section 10.1. Section 10.2 discusses the main contribution of this thesis, i.e. the novel coordination approach through task allocation, based on social networks. Section 10.3 discusses a number of future directions for future research, following from this thesis.

10.1 INDABA

The first part of this thesis investigated agent architectures and multi-agent architectures, with the emphasis on a particular type of agent, namely robots.

Chapter 3 provided an overview of three major robot architectures. Each of the architectures, namely reactive, symbolic and hybrid, were first discussed in general terms, followed up by a more detailed discussion of an example of each architecture.

An overview of two major multi-robot team architectures was presented in chapter 4. Again, each architecture was first discussed in general terms, followed by a more detailed discussion of a particular example of each architecture.

Based on the findings of the study of agent architectures, as presented in chapters 3 and 4, a new architecture was proposed for the development of cooperative multi-robot teams. The new architecture, INDABA, was introduced in chapter 5. INDABA is a conceptual framework and guideline for the agents' implementation, rather than a fully developed and prescriptive framework. This allows the architecture to be applied to a variety of robotic platforms. Although INDABA is not prescriptive with respect to technologies, particular implementations and coordination mechanisms, INDABA is prescriptive in the adopted layering approach.

Most of the current robot architectures used hybrid robot architectures that consist of three layers. INDABA consists of four layers, with the fourth layer added to facilitate ease of implementation of coordination mechanisms.

The INDABA framework was used to develop an abstract robot simulator and simulated robot environment. The abstract robot simulator was discussed in chapter 7 and the simulated robot environment was introduced in chapter 8. In both examples, the INDABA framework was flexible enough to cater for different levels of abstractions used by the simulators, as well as to cater for different coordination mechanisms.

A full implementation of all four-layers of the INDABA framework in a physical environment with robots was described in chapter 9.

INDABA has proved to be a suitable architecture for implementing embedded agents, namely robots, either in simulated or in physical environments. The addition of the fourth layer, the interaction layer, facilitated the implementation of a coordination mechanism, for example the auctioning mechanism and the social networks based mechanisms.

10.2 The Social Networks Based Approach

The main contribution of this thesis is the development of a flexible, biology inspired approach to coordination through the use of social networks.

Various existing coordination approaches to multi-robot team task allocation were overviewed in chapter 6. Social networks and related concepts were also introduced in chapter 6. A novel coordination mechanism for multi-robot teams, based on social networks, was also presented in chapter 6. This new, social networks based coordination mechanism was tested in the experiments in the following chapters.

The social networks based approach was presented in great detail and a comparison with a natural system was made to illustrate its biological and societal origins.

The social networks based approach was first tested using the abstract robot simulator. The abstract robot simulator provided for simulations with a relatively large number of agents (a population of fifty agents was created) over a relatively large number of simulated tasks executions (ranging from 50 to 700). Probabilistic team selection was considered and rejected in favour of straightforward ranking selection, as it did not perform as well as the straightforward ranking selection.

The social networks based approach consistently performed better than a pure market based approach in conditions of uncertainty about task details. Furthermore, the new approach exhibited excellent learning capacity.

The social networks based approach exhibited an intriguing similarity between the overall behaviour of the multi-robot society and biological systems. Cliques emerged, as well as natural specialisation of agents toward particular tasks. The importance of kinship and trust were confirmed, even in artificial agent societies. Furthermore, the social networks approach has proven that concepts such as kinship and trust, traditionally related to higher mammalian societies, can be used for coordination of artificial societies of agents.

Results from a simulated robot environment, using the same social networks based approach to coordination, followed up the results from the simulations done in the abstract robot simulator. The results were similar to the results from the simulations done in the abstract robot simulator. The social networks based approach was yet again confirmed to be valid.

10.3 Directions for the Future Research

A number of aspects have been identified that merit further research. These are summarised next.

10.3.1 Use of Multiple Alternative Coordination Methods in INDABA

Coordination through task allocation is not the only coordination method and the consequence of other coordination methods such as negotiation [140] and cooperation should also be explored. A possible direction for further research is to implement various coordination methods in the interaction layer of INDABA, and to build a mechanism that will choose the most appropriate one for the current task and for the society of agents. This possibility can be seen as a negotiation protocol between agents. In other words, based on agents' capabilities, a consensus is reached on which coordination mechanism to use.

10.3.2 Flexible Information Exchange in Multi-robot Teams

For the purpose of this thesis, the information that is exchanged between the robots in a multi-robot team was predefined and its format was hard-coded (such as information about position of food and environment). Ideally, robots should be able to discover new concepts, and share these concepts and knowledge about the concepts with other team members. For example, new environment attributes could be detected that were not pre-defined. For this purpose, future applications should consider use of more flexible mechanisms, based for example on KQML [66][99] and XML [186] for information and knowledge exchange.

10.3.3 Investigation of Applicability of Additional Social Relationships to Multi-robot Systems

In this thesis, only two social relationships, kinship and trust, were used for the implementation of the social networks based approach to coordination. In the real-world, more social relationships exist among human society. For example, by living in a specific area, working in a particular environment etc. One social relationship that easily comes to mind as a potential candidate for application in multi-robot teams is that of friendship. Friendship could be implemented around the concept of reciprocal altruism [203]. The mechanism for maintenance of social relationship can also benefit

from further research by investigating a partial goal completion reward system, as opposed to the currently implemented "all or nothing" approach.

10.3.4 Social Networks as a Rule-Extraction Mechanism

The learning capacity of the social network based approach opens interesting possibilities that can be explored in future work. In the INDABA deliberator layer, all robot and environment attributes are defined in almost symbolic terms. Using such definitions, it should be relatively possible to express selection rules in a symbolic form, for example in the form of production rules. Based on the best performing scout attributes (refer to section 7.5.2), it would be simple to extract a rule for scout selection, e.g.

IF SPEED_FAST **AND** DETECTION_ADVANCED **AND** POWER_BATTERY **THEN SCOUT.**

Future research will develop a mechanism to extract such symbolic rules from social networks.

10.3.5 Investigation into a More Formal Kinship Rating Mechanism

The current implementation of kinship rating is fairly crude and heuristic. A different, more formal mechanism for determining the strength of kinship relationship should be investigated. A possible direction for research is to expand on work that proposes encoding of robot building blocks in a formal way, for example, using a graph grammar as in [147]. To illustrate the point, the research done for this thesis found that the sensitivity of the used robot platform to sensor positioning was somewhat of a surprise. Kinship rating, as currently implemented, takes into account only the existence of a sensor, not sensor positioning. The sensor positioning influences sensor readings and the kinship relationship should take this into account.

Bibliography

1) G. Agha. *ACTORS: A Model of Concurrent Computation in Distributed Systems*. MIT Press, 1986.

2) K. Altenburg. *Adaptive Resource Allocation for a Multiple Mobile Robot Systems using Communication*. Technical Report, NDSU-CSOR-TR-9404, North Dakota State University, 1994.

3) J. Ambros-Ingerson and S. Steel. Integrating Planning, Execution and Monitoring. *In Proceedings of the Seventh National Conference on Artificial Intelligence*, AAAI-88, pp. 83-88.

4) J. R. Anderson. *Rules of the Mind*. Lawrence Erlbaum Associates, 1993.

5) A. Andres Perez-Uribe and B. Hirsbrunner. Learning and Foraging in Robot-bees. In Meyer, Berthoz, Floreano, Roitblat and Wilson (eds.), *SAB2000 Proceedings Supplement Book*, International Society for Adaptive Behavior, Honolulu, pp. 185-194, 2000.

6) R. C. Arkin. Integrating Behavioural, Perceptual and World Knowledge in Reactive Navigation, In *Robotics and Autonomous Systems 6*, pp. 105-122, 1990.

7) M. Asaka, A. Taguchi and S. Goto. The Implementation of IDA: An Intrusion Detection Agent System, In *Proceedings of the 11th Annual FIRST Conference on Computer Security Incident Handling and Response* (FIRST'99). pp. 13-24, 1999.

8) R. Axelrod. *The Evolution of Cooperation* Basic Books, 1984.

9) R. Aylett, A. M. Coddington, D. P. Barnes and R. A. Ghanea-Hercock. What Does a Planner Need to Know About Execution? In *Recent Advances in AI Planning, 4th European Conference on Planning*, Toulouse, France, pp. 26-38,1997.

10) R. Aylett, A.M. Coddington., D.P. Barnes and R.A. Ghanea -Hercock What does a planner need to know about execution? In S. Steel and R. Alami (eds.) *Recent advances in AI planning,* Springer, pp. 26-38, 1997.

11) R. Aylett, R. A. Ghanea-Hercock and A. M. Coddington. Supervising multiple cooperating mobile robots. In *Proceedings of the first*

international conference on Autonomous agents, pp.514-515, Marina del Rey, California, United States, 1997.

12) T. Balch. Hierarchic Social Entropy: An Information Theoretic Measure of Robot Group Diversity, In *Autonomous Robots* 8(3), pp. 209-238, 2000.

13) T. Balch. Learning roles: Behavioural diversity in robot teams. In *AAAI-97 Workshop on Multiagent Learning*, Providence, 1997.

14) T. Balch Measuring Robot Group Diversity. In T. Balch T, L. E. Parker (eds.) *Robot teams*, A K Peters Ltd, pp. 93-135, 2002.

15) D. P. Barnes and J. O. Gray. Behaviour Synthesis for Co-operant Robot Control, In *Proceedings IEE International Conference Control 91*, pp. 1153-1140, 1991.

16) D. P. Barnes, R.A. Ghanea-Hercock, R.S. Aylett and A. Coddington. Many hands make light work? An investigation into behaviourally controlled co-operant autonomous mobile robots. In *Proceedings of 1st International Conference on Autonomous Agents.* Marina del Rey, pp. 413 - 420, 1997.

17) D. P. Barnes. A Behaviour Synthesis Architecture for Co-operant Mobile Robots. In J. O. Gray and D. G. Caldwell (eds) *Advanced Robotics & Intelligent Machines*, IEE Control Engineering Series 51, pp. 295-314, 1996.

18) J. Bates, A. Loyall and W. S. Reilly. *An Architecture for Action, Emotion and Social Behavior.* Technical Report CMU-CS-92-144, School of Computer Science, Carnegie Mellon University, 1992.

19) R. Beckers, O.E. Holland and J.L Deneubourg. From local actions to global tasks: stigmergy and collective robotics. In R. Brooks and P. Maes(eds) *Proceedings of fourth international workshop on artificial life,* MIT Press, Boston, pp.181-189, 1994.

20) D. Beetham. Models of Bureaucracy. *Bureaucrac*y, pp. 9-47, 1987.

21) B. M. Bloomberg, P.M. Todd and P. Maes. No Bad Dogs: Ethological Lessons for Learning in Hamsterdam, In *From Animals to Animats, Proceedings of the Fourth International Conference on Adaptive Behaviour*, pp. 295-304, 1996.

22) R. P. Bonasso, R. J. Firby, E. Gat, D. Kortenkamp, D. P. Miller and M. G. Slack. Experiences with an Architecture for Intelligent Reactive Agents. In *Journal of Experimental and Theoretical AI 9(2)*, 1997.

23) R. P. Bonasso. Integrating Reaction Plans and Layered Competences Through Synchronous Control. In *Proceedings of the International Joint Conference on Artificial Intelligence,* pp. 1225-1233,1991.

24) A. H. Bond and L. Gasser. An Analysis of Problems and Research in DAI. In A. H. Bond and L. Gasser (eds) *Readings in Distributed Artificial Intelligence*, pp. 3-35, Morgan Kaufmann Publishers, San Mateo CA, 1988.

25) G. N. Boone. Efficient Reinforcement Learning: Model-Based Acrob ot Control. In *IEEE International Conference on Robotics and Automation,* pp. 229-234, Albuquerque 1997.

26) V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology*, MIT Press, 1984.

27) R. A. Brooks and J. H. Connell. Asynchronous Distributed Control System for a Mobile Robot. *SPIE Vol 727 Mobile Robots*, pp. 77-84, 1986.

28) R. A. Brooks, Intelligence Without Representation, In *Artificial Intelligence Journal* 47, pp. 139–159, 1991.

29) R. A. Brooks, J. H. Connell, P. Ning and Herbert. *A Second Generation Mobile Robot.* Technical Report MIT-AIM-1016, 1988.

30) R. A. Brooks. A Robot That Walks: Emergent Behaviour from a Carefully Evolved Network. *Neural Computation 1*, pp. 153-162, 1989.

31) R. A. Brooks. A robust layered control system for a mobile robot. In *IEEE Transactions on Robotics and Automation,* 2(1), pp. 14-23, April 1986.

32) R. A. Brooks. Elephants Don't Play Chess. In P. Maes (ed) *Designing Autonomous Agents* pp. 3-15, MIT Press, 1990.

33) R. A. Brooks. Coherent Behaviour from Many Adoptive Processes, In *From Animals to Animats. Proceedings of the Third International Conference on Adaptive Behaviour*, pp. 421-430, 1994.

34) R. A. Brooks. How to build Complete Creatures Rather than Isolated Cognitive Simulators, in K. VanLehn (ed.), *Architectures for Intelligence*, pp. 225-239, Lawrence Erlbaum Associates, 1991.

35) R. A. Brooks. Intelligence without reason. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence* (IJCAI-91), pp. 569-595, Sydney, Australia, 1991.

36) B. Burmeister, A. Haddadi, and G. Matylis. Applications of multi-agent systems in traffic and transportation. In *IEE Transactions on Software Engineering*, 144(1), pp. 51-60, 1997.

37) M. Busuioc. Distributed Intelligent Agents – A Solution for the Management of Complex Services. In *The Intelligent Agents for Telecom Applications Workshop Proceedings*, Budapest, 1996.

38) Y. U. Cao, A. S. Fukunaga and A. B. Kahng. Cooperative Mobile Robotics: Antecedents and Directions. *Autonomous Robots 4*, pp. 1-23, 1997.

39) K. Chapek . *Rossum's Universal Robots,* Penguin Books, reprint 2004, original published 1921.

40) A. Chavez and P. Maes. Kasbah: An agent marketplace for buying and selling goods. In *proceedings First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*. Pp. 75-90, 1996.

41) J. Chu-Carroll and S. Carberry. Conflict Detection and Resolution in Collaborative Planning. *Agent Theories, Architectures, and Languages*, II, Springer-Verlag Lecture Notes in Computer Science, pp. 111-126, 1996.

42) R. Collins *Theoretical Sociology*, New York, Basic Books, 1988.

43) J. H. Connell. Creature Building with the Subsumption Architecture, In *International Journal of Computing and AI*, pp. 1124-1126, 1987.

44) J. H. Connell. Minimalist Mobile Robotics: A Colony Style Architecture for an Artificial Creature, *Perspectives in Artificial Intelligence 5,* Academic Press, 1990.

45) J. H. Connell. SSS: A Hybrid Architecture Applied to Robot Navigation. In *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 2719-2724, 1992.

46) S. Conry, R. Meyer and V. Lesser. *Multistage Negotiation in Distributed Planning*. COINS Technical Report, University of Massachusetts, 1986.

47) K. Dautenhahn. Biologically inspired robotic experiments on interaction and dynamic agent-environment couplings. In *Proceedings of Workshop*

SOAVE'97, Selbstorganization von Adaptivem Verhalten, Ilmenau, pp. 23-28, 1997.

48) K. Dautenhahn. Embodiment and Interaction in Socially Intelligent Life-Like Agents. *Lecture Notes in Computer Science*, Vol. 1562, pp. 102-142, 1999.

49) R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20, pp. 63-100, 1983.

50) G. Dedeoglu, G. Sukhatme and M. J. Matarić. Incremental On-Line Topological Map Building for a Mobile Robot. In *Proceedings, Mobile Robots XIV- SPIE*, pp. 129-139, 1999.

51) M. B. Dias and A. Stentz. A Free Market Architecture for Distributed Control of a Multirobot System. In *Proceedings of 6th International Conference on Intelligent Autonomus Systems*, pp. 115-122, 2000.

52) F. Dignum, B. Dunin-Keplicz and R. Verbrugge. Agent Theory for Team Formation by Dialogue, In *Proceedings of the 7th International Workshop on Intelligent Agents VII* , pp. 150-166, 2000.

53) M. d'Inverno and M. Luck . Sociological Agents for Effective Social Action. In *Proceedings of Proceedings of the Fourth International Conference on Multi-Agent Systems*, pp. 379-380, Boston, 2000.

54) M. d'Inverno, M. Fisher, A. Lomuscio, M. Luck, M. de Rijke, M. Ryan and Wooldridge. Formalisms for Multi-Agent Systems. In *Knowledge Engineering Review,* 12(3), pp. 315-321, 1997.

55) J. Doran and M. Palmer. The EOS project: integrating two models of {Palaeolithic} social change, in N. Gilbert and R. Conte (eds), *Artificial Societies* London: UCL Press, pp. 103-125, 1995.

56) G. Dudek, M. R. M. Jenkin, E. Milios and D. Wilkes. A Taxonomy for Multi-Agent Robotics. *Autonomous Robots 3 (4)*, pp. 375-397, 1996.

57) G. Dudek, M. R. M. Jenkin, E. Milios and D. Wilkes. A Taxonomy for Swarm Robots. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems,* pp. 441-447, 1993.

58) B. Dunin-Keplicz and J. Treur. Compositional Formal Specification of Multiagent Systems. In *Intelligent Agents, Volume 890 of Lecture Notes in Artificial Intelligence*, pp. 102-117, Springer, 1994.

59) E. H. Durfee. *Coordination of Distributed Problem Solvers*. Kluwer Academic Publishers, 1988.

60) C. Elsaesser and M. G. Slack. Integrating Deliberative Planning in a Robot Architecture. *In Proceedings of the AIAA/NASA Conference on Intelligent Robots in Field, Factory, Service and Space,* (CIRFFSS '94), Houston, Texas, pp. 782–787, 1994.

61) A. P. Engelbrecht. *Computational Intelligence, An Introduction,* Wiley Publishers, 2002.

62) O. Etzioni, N. Lesh and R. Segal. Building Softbots for UNIX. *Software Agents – Papers from the 1994 Spring Symposium*, pp. 9-16, AAAI Press, 1994.

63) I. A. Ferguson. *Touring Machines: An Architecture for Dynamic, Rational, Mobile Agents*. PhD thesis, University of Cambridge, 1992.

64) M. Ferrari, G. Ferrari and R. Hempel. *Building Robots with Lego Mindstorms*, Syngress, 2002.

65) R. Fikes and N. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, In *Artificial Intelligence*, 2 (3), pp. 189-208, 1971.

66) T. Finin, Y. Labrou and J. Mayfield. KQML as an Agent Communication Language. *Software Agents*, *MIT Press*, Cambridge, pp. 291-316,1997.

67) R. J. Firby. *Adaptive Execution in Complex Dynamic Worlds*. PhD thesis, Yale University, 1989.

68) R. J. Firby. *Adaptive Execution in Dynamic Domains*. Technical Report *YALEU/CSD/RR#672*, Yale University, 1989.

69) T. Fong, I. Nourbakhsh and K. Dautenhahn. A Survey of Socially Interactive Robots. In *Robotics and Autonomous Systems 42*, pp. 143–166, 2003.

70) L. Gasser. Social Conceptions of Knowledge and Action: DAI foundations and Open Systems Semantics. In *Artificial Intelligence, 47* pp. 107-138, 1991.

71) E. Gat. Integrating Planning and Reaction in a Heterogeneous Asynchronous Architecture for Controlling Mobile Robots. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 809-815, 1992.

72) E. Gat. On Three-Layer Architectures. In D. Kortenkamp, R. P. Bonasso and R. Murphy (eds.) *Artificial Intelligence and Mobile Robots* AAAI Press, 1998.

73) E. Gat. *Reliable Goal-directive Reactive Control for Real-World Autonomous Mobile Robots*. PhD Thesis, Virginia Polytechnic Institute and State University, Blacksburg, 1991.

74) E. Gat. *Three-layer architectures, Artificial intelligence and mobile robots: case studies of successful robot systems*, MIT Press, Cambridge, MA, 1998.

75) M. Genesereth and R. Fikes. *Knowledge Intechange Format*. *Reference Manual Version 3.0,* Technical Report, Computer Science Department, Stanford University, 1992.

76) B. P. Gerkey and M. J. Matarić. A Framework for Studying Multi-Robot Task Allocation In Multi-Robot Systems. In A.C. Schultz and others (eds.), *From Swarms to Intelligent Automata*, Vol 2, Kluwer Academic Publishers, pp. 15-26, 2003.

77) B. P. Gerkey and M. J. Matarić. Sold!: Auction Methods for Multirobot Coordination. In *IEEE Transactions on Robotics and Automation*, 18 (5), pp. 758-768, 2002.

78) G. Gilart, R. Chatila and M. Vaisset. An Integrated Navigation and Motion Control System for Multisensory Robots. In *Robotic Research 1*, pp. 191-214, MIT Press, 1984.

79) D. E. Goldberg. *Genetic Algorithms in Search Optimisation and Machine Learning*, Addison-Wesley, 1989.

80) S. Green, L. Hurst, B. Nangle, P. Cunnigham, F. Somers and R. Evans. *Software Agents: A Review*, Trinity College Dublin1 Broadcom E'ireann Research Ltd.2, 1997.

81) M. Griss and G. Pour. Accelerating Development with Agent Components. In *IEEE Computer*, 34(5), pp. 37-43, 2000.

82) B. J. Grosz and R. Davis. AAAI Report to ARPA on 21st Century Intelligent Systems, In *AI Magazine*, pp. 10-20, 1994.

83) C. Guilfoyle. Vendors of Agent Technology. In *UNICOM Seminar on Intelligent Agents and their Business Applications*, pp. 135-142, 1995.

84) R. Hartley and F. Pipitone. Experiments with the Subsumption Architecture. In *Proceedings of 1991 IEEE International Conference on Robotics and Automation*, v2, pp. 1652-1658, IEEE Computer Society Press, 1991.

85) J. K. Hodgins and D.C. Brogan. Robot Herds: Group Behaviour from Systems with Significant Dynamics, Artificial Life IV. In *Proceeding of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, 1994.

86) C. A. Iglesias, J. Centeno-González and J. R. Velasco. MIX: A General Purpose Multiagent Architecture. In *ATAL 1995*, pp. 251-266, Montreal, Canada, 1995.

87) P. Jackson. *Introduction to Expert Systems.* Addison-Wesley Publishers, 1990.

88) N. R. Jennings, K. Sycara and M. Wooldridge. A Roadmap of Agent Research and Development, In *Autonomous Agents and Multi-Agent Systems 1*, pp. 275-306, 1998.

89) S. Johnson. *Emergence: The Connected Lives of Ants, Brains, Cities and Software.* Scribner, 2001.

90) C. G. Jung and K. Fischer. *Methodological comparison of agent models,* Technical Report RR-98-1, DFKI GmbH, Saarbrücken, Germany, 1998.

91) L. P. Kaebling, M. L. Littman and A. W. Moore. Reinforcement Learning: A Survey. In *Journal of Artificial Intelligence Research*. Vol 4, pp. 237-285, 1996.

92) L. P. Kaelbling. A Situated Automata Approach to Design of Embedded Agents. In *SIGART Bulletin*, 2 (4), pp. 85-88, 1991.

93) P. Kearney, A. Sehmi, and R. Smith. Emergent Behaviour in a Multi-agent Economics Simulation. In A. G. Cohn, (ed) *Proceedings of the 11th European Conference on Artificial Intelligence*. John Wiley, 1994.

94) R. Khosla and T. Dillon. *Engineering Intelligent Hybrid Multi-Agent Systems* Kluwer Academic Publishers, 1997.

95) D. Kirsh. Today the Earwig, Tomorrow Man? *Artificial Intelligence 47,* pp. 161-184, 1991.

96) H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa and H. Matsubara. RoboCup: A Challenge Problem for AI. In *AAAI*, *AI Magazine*, pp. 73-85, Spring 1997.

97) W. Kohler. *The Mentality of Apes*, New York, 1925.

98) R. C. Kube and E. Bonabeau. Cooperative transport by ants and robots, In *Robotics and Autonomous Systems,* pp. 85-101, 2000.

99) Y. Labrou and T. Finin. *A Proposal for a new KQML Specification*, Technical Report TR CS-97-03, University of Maryland Baltimore County, Baltimore, MD 21250, 1997.

100) S. Lalande, A. Drogoul, and B. Thierry. MACACA: a Multi-Agent Computer simulation of Animal Communities based on Alliances. In *Proceedings of Simulating Societies Symposium*, 1995.

101) F. Lehmann (ed). Semantic Networks. *Part of International Series in Modern Applied Mathematics and Computer Science* 24, 1992

102) R. Levacic. Markets and Governments. *The Coordination of Social Life,* (eds.) G. Thompson, J. Frances, Levačić and J. Mitchell, Sage Publications, London, pp. 45-47 ,1991.

103) P. Lima, R. Ventura, A. Aparicio and L. Custodio. A functional architecture for a team of fully autonomous cooperative robots. *RoboCup*, pp. 378-389, 1999.

104) M. Ljunberg and A. Lucas. The OASIS Air Traffic Management System. In *Proceedings of the Second Pacific Rim International Conference on AI*, Seoul, 1992.

105) P. Maes. Agents that Reduce Work and Information Overload. *Communications of the ACM*, 37 (7), pp. 31-40, 1994.

106) P. Maes. The Agent Network Architecture (ANA). In *SIGART Bulletin*, 2 (4), pp. 115-120, 1991.

107) M. J. Matarić, M. Nilsson and K. Simsarian, In *Proceedings IROS-95*, Pittsburgh, pp. 556-561, 1995.

108) M. J. Matarić. Behaviour Based Control: Examples from Navigation, Learning and Group Behaviour. In *Journal of Experimental and Theoretical Artificial Intelligence*, *Special Issue on Software Architectures for Physical Agents*, 9 (2-3), pp. 323-336, 1997.

109) M. J. Matarić. Behaviour Based Robotics. *MIT Encyclopaedia of Cognitive Science*, MIT Press, pp74-77, 1999.

110) M. J. Matarić. Designing and Understanding Adaptive Group Behaviour. In *Adaptive Behaviour* 4 (1), pp. 51-80, 1995.

111) M. J. Matarić. Great Expectations: Scaling Up Learning by Embracing Biology and Complexity, NSF Workshop on Development and Learning, Michigan State University, 2000.

112) M. J. Matarić. Integration of Presentation Into Goal Driven Behaviour Based Robots. In *IEEE Transactions on Robotics and Automation,* 8 (3), pp. 304-312, 1992.

113) M. J. Matarić. *Interaction and Intelligent Behaviour*, PhD Thesis, MIT, 1994.

114) M. J. Matarić. Issues and approaches in design of collective autonomous agents. In *Robotics and Autonomous Systems*, *16*, pp. 321-331, 1995.

115) M. J. Matarić. Learning in Behaviour Based Multi Robot Systems. In *Policies, Models and Other Agents*, *Special issue on Multi-disciplinary studies of Multi-agent Learning*, 2 (1), pp. 81-93, 2001.

116) M. J. Matarić. Learning Social Behaviour. In *Robotics and Autonomous Systems 20*, pp. 191-204, 1997.

117) M. J. Matarić. Learning to Behave Socially, In *Proceedings From Animals to Animats 3.* Third European Conference on Artificial Life, pp. 453-462, 1994.

118) M. J. Matarić. Reinforcement Learning in the Multi-Robot Domain, In *Autonomous Robots,* 4 (1), pp. 73-83, 1997.

119) M. J. Matarić. Using Communication to Reduce Locality in Distributed Multi-Agent Learning. In *Journal of Experimental and Theoretical Artificial Intelligence, special issue on learning in DAI systems*, 10 (3), pp. 357-369, 1998.

120) P. McCorduck. *Machines Who Think*, *A Personal Inquiry into the History and Prospects of Artificial Intelligence*, W.H.Freeman, 1979.

121) D. McFarland. Towards Robot Cooperation, In *From Animals to Animats. Proceedings of the Third International Conference on Adaptive Behaviour*, pp. 440-444, 1994.

122) J. D. McLurkin. *Using Cooperative Robots for Explosive Ordnance Disposal*, MIT Research Report for Naval EOD Technical Division, 1997.

123) L. D. Mech. *The Wolf,* University of Minnesota Press, 1970.

124) F. Michaud and M. J. Matarić. Representation of Behavioural History for Learning in Non-stationary Conditions. In *Robotics and Autonomous Systems 29*, pp.187-200, 1999.

125) J. C. Mitchell. The concept and use of social networks. *Social networks in urban situations*, Manchester University Press, Manchester, 1969.

126) H. P. Moravec. Locomotion, Vision and Intelligence. *Robotics Research 1*, MIT Press, pp. 215-224, 1984.

127) H. P. Moravec. The Stanford Cart and CMU Rover. In I. J. Cox and G. T. Wilfong (eds.) *Autonomous Robot Vehicles,* Springer-Verlag, pp. 407-441, 1990.

128) A. Mukerjee and A. D. Mali. Agent Models of Intelligence – Limitations and Prospects, In G. Parker (ed) *Proceedings ISORA-98*, Alaska 1998.

129) J. P. Müller and P. Jörg. The Design of Intelligent Agents, A Layered Approach, In *Series: Lecture Notes in Computer Science, Subseries: Lecture Notes in Artificial Intelligence,* Vol. 1177, 1996, XV, pp. 227, 1996.

130) J. P. Muller, M. Pishel and M. Thiel. Modelling Reactive Behaviour in Vertically Layered Agent Architectures: Intelligent Agents. *Lecture Notes in Artificial Intelligence 890*, pp. 261-276, 1995.

131) R. R. Murphy. Trial by Fire. *IEEE Robots & Automation* (11):3, pp. 50-61. 2004.

132) R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator and W. Swartout. Enabling Technology for Knowledge Sharing, In *AI Magazine 12 (3)*, pp. 36-56, 1991.

133) A. Newell and H. A. Simon. Computer Science as Empirical Enquiry. In *Communications of the ACM 19*, pp. 113-126, 1976.

134) A. Newell and H.A. Simon. GPS, a Program That Simulates Human Thought. *Computers and Thought,* MIT Press, USA, pp. 279-293, 1995.

135) A. Newell. Unified Theories of Cognition, *For Artificial Intelligence*, 59, (1-2), pp. 285-294. Reprinted in Clancey, Smoliar and Stefik (eds.)

Contemplating Minds: *A Forum for Artificial Intelligence*, MIT Press, 1993.

136) N. J. Nilsson (ed). *Shakey the Robot*, Technical Report, SRI AI Center, 1984.

137) N. J. Nilsson. *Towards agent programs with circuit semantics*. Technical Report STAN-CS-92-1412, Computer Science Department, Stanford University, Stanford, 1992.

138) H. S. Nwana. Software Agents: An Overview. *Knowledge Engineering Review,* 11(3), pp. 1-40, 1996.

139) Object Management Group – Agent Platform Special Interest Group. *Agent Technology – Green Paper Version 1*, 2000.

140) P. Panzarasa and N. R. Jennings. Social Influence, Negotiation and Cognition, In *Simulation Modelling Practice and Theory* 10(5), pp. 417-453, 2002.

141) P. Panzarasa and N. R. Jennings. The organisation of sociality: a manifesto for a new science of multi-agent systems. In *Proceedings 10th European Workshop on Multi-Agent Systems (MAAMAW-01)*, Annecy, France, 2001.

142) L. E. Parker, C. Touzet and D. Jung. Learning and Adaptation in Multi-Robot Teams. In *Proceedings of Eighteenth Symposium on Energy Engineering Sciences*, pp. 177-185, 2000.

143) L. E. Parker, C. Touzet anf F. Fernandez. Techniques for learning in multi-robot teams. In T. Balch T, L. E. Parker (eds.) *Robot teams*, A K Peters Ltd, pp. 191-237, 2002.

144) L. E. Parker. ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation. In *IEEE Transactions on Robotics and Automation,* 14 (2), pp. 220-240, 1998.

145) L.E. Parker. L-ALLIANCE: Task-Oriented Multi-Robot Learning in Behaviour Based Systems. In *Journal of Advanced Robotics* 11(4), pp. 305-322, 1997.

146) J. C. Penberthy and D. Weld. UCPOP: A Sound, Complete, Partial-Order Planner for ADL. *Proceedings 3rd International Conference on Principles of Knowledge Representation and Reasoning*, pp. 103-114, 1992.

147) M. Peysakhov, V. Galinskaya and W. C. Regli. Using Graph Grammars and Genetic Algorithms to Represent and Evolve Lego Assemblies, In D/ Whitley (ed) *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, pp. 269-276, 2000.

148) R. Pfeifer and C. Scheier. *Understanding Intelligence*. MIT Press, 1999.

149) S. Picault. A Multi-Agent Simulation of Primate Social Concepts. In *Proceedings European Conference on Artificial Intelligence*, pp. 327-328, 1998.

150) G. Pour. Internet-Based Multi-Mobile-Agent Framework for Planetary Exploration, In *Proceedings of International Conference on Intelligent Agents, Web Technology and Internet Commerce,* pp. 153-163, 2001.

151) A. Rao and M. Georgeff. BDI Agents: From Theory to Practice. In *Proceedings of the First International Conference on Multi-Agent Systems*, San Francisco, USA, pp. 312-319, 1995.

152) A. Rao and R. Georgeff. Modelling Rational Agents Within a BDI-Architecture. In *Proceeding of Knowledge Representation and Reasoning,* pp. 473-484, San Mateo CA, Morgan Kaufmann Publishers, 1991.

153) B. Raphael. *The Thinking Computer: Mind Inside Matter*, 1976. *Robotics and Autonomous Systems*. (30):2 , pp. 85-101, 2000.

154) D. Rodić and A. P Engelbrecht. Investigation into applicability of social networks as a task allocation tool for multi-robot teams. *Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003)*, Singapore, Program and Abstracts, pg. 133, 2003.

155) D. Rodić and A. P Engelbrecht. Investigation of Low Cost Hybrid Three-Layer Robot Architecture. *Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003),* Singapore, Program and Abstracts, pg. 85, 2003.

156) D. Rodić and A. P Engelbrecht. Social Networks as a Coordination Technique for Multi-Robot Systems. *Intelligent Systems Design and Applications*, Springer, pp. 503-513, 2003.

157) D. Rodić and A. P. Engelbrecht. INDABA – Proposal for Intelligent Distributed Agent Based Architecture. *Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003)*, Singapore, Program and Abstracts, pg 85, 2003.

158) D. Rodić and A. P. Engelbrecht. Social Networks as a Task Allocation Tool for Multi-Robot Teams. *South African Computer Science Journal 33,* pp. 53-67, 2004.

159) J. K. Rosenblatt and D. W. Payton. A fine-grained alternative to the subsumption architecture for mobile robot control. *In Proceedings of the IEEE International Conference on Neural Networks 2*, pp. 317-324, 1989.

160) J. S. Rosenschein and G. Zlotkin. *Rules of encounter: designing conventions for automated negotiation among computers*, MIT Press, 1994.

161) T. Sandholm and V. Lesser. Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Protocol. In *Proceedings of First International Conference on Multiagent Systems (ICMAS95*), San Francisco, AAAI Press and MIT Press, pp. 328-335, 1995.

162) R. Sargent, M. Resnick, F. Martin and B. Silverman. Building and Learning with Programmable Bricks. In the *Logo Update*, (3):3, 1995.

163) R. D. Sarvapali, D. Huynh and N. R. Jennings. Trust in Multi-Agent Systems, In *The Knowledge Engineering Review,* 2004.

164) M. Schillo, P. Funk and M. Rovatsos. Using Trust for Detecting Deceptive Agents in Artificial Societies. In *Applied Artificial Intelligence, Special Issue on Trust, Deception and Fraud in Agent Societies,* 14(8), pp. 825-848, 2000.

165) M. Schoppers. Universal Plans for Reactive Robots in Unpredictable Domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1039-1046, 1987.

166) C. Scott, Y. Labrou and T. Finin. Coordinating Agents using Agent Communication Languages Conversations. *Coordination of Internet Agents: Models, Technologies, and Applications*, pp. 183-196, 2001.

167) J. Scott. *Social Network Analysis*. Sage Publications, 1991.

168) S. Sen, M. Sekaran and J. Hale. Learning to Coordinate without Sharing Information. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, pp. 426- 431, 1994.

169) M. Shelly. *Frankenstein (Changing Our World).* Bantam (reprint edition) 1984.

170) R. Shinghal. *Formal Concepts in Artificial Intelligence*, Chapman & Hall, 1992.

171) E. H. Shortliffe. *Computer-Based Medical Consultations, MYCIN,* New York, Elsevier, 1976.

172) M. Sierhuis, M.H. Sims, W. J. Clancey and P. Lee. Applying Multiagent Simulation to Planetary Surface Operations. In L. Chaudron (Ed) *COOP'2000 Workshop on Modelling Human Activity* pp. 19-28 Sophia Antipolis, France, 2000.

173) R. Simmons and D. Apfelblaum. A Task Description Language for Robot Control, In *Proceedings Conference on Intelligent Robotics and Systems,* 1998.

174) K. T. Simsarian and M. J. Matarić. Learning to Cooperate Using Two Six Legged Mobile Robots. In *Proceedings, Third European Workshop of Learning Robots*, Heraklion, Crete, Greece, 1995.

175) R. G. Smith. The Contract Net Protocol: High-Level Communication and Control in Distributed Problem Solver. In *IEEE Transactions on Computers,* C-29 (12), 1980.

176) A. Smith. *Wealth of the Nations*, Prometheus Books, 1991, original published in 1776.

177) L. Steels . Between Distributed Agents through Self-Organisation. In Y Demazeau and J-P Müller (eds) *Decentralized AI*, Elsevier Science Publishers B.V, pp. 175-196, 1990.

178) L. Steels. The Artificial Life Roots of Artificial Intelligence. In *Artificial Life 1*, (1), pp. 75-110, 1994.

179) T. Tassier and F. Menczer. Emerging Small-World Referral Networks in Evolutionary Labor Markets, In *IEEE Transactions on Evolutionary Computation (Special Issue on Computational Economics), 5 (5),* pp. 482-492, 2001.

180) G. Thompson. Markets, Hierarchies & Networks. *The Coordination of Social Life,* (eds.) G. Thompson, J. Frances, Levačić and J. Mitchell, Sage Publications, London, 1991.

181) V. Tsvetovatyy and M. Gini. Toward a Virtual Marketplace. In *Proceedings of Practical Application of Intelligent Agents and Multi-Agent Technology* '96, London, pp. 596-612, 1996.

182) URL www.baumfamily.org/nqc/, accessed July 2003.

183) URL : http://members.cox.net/pbrick-alpha/, accessed July 2003.

184) URL : www.k-team.com, accessed February 2004.

185) URL : www.mindstorms.com, accessed June 2003.

186) URL : www.w3.org/XML, accessed September 2004.

187) URL: http://marsrovers.jpl.nasa.gov/, accessed August 2004.

188) URL: http://www.pantheon.org/articles/r/rabbi_loeb.html, accessed August 2004.

189) URL: www.ai.mit.edu/projects/ants, accessed July 2003.

190) URL: www.Amazon.com, accessed August 2003.

191) URL: www.astronautix.com, accessed March 2004.

192) URL: www.google.com, accessed February 2002.

193) S. Vere and T. Bickmore. A Basic Agent. *Computational Intelligence,* Vol 6, pp. 41-60, 1990.

194) R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras and H. Das. The CLARAty Architecture for Robotic Autonomy. In *Proceedings of the 2001 IEEE Aerospace Conference*, Big Sky, Montana, 2001.

195) K. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*, Cambridge University Press, 1994.

196) B. B. Werger and M. J. Matarić. Broadcast of local eligibility for multi-target observation in L. E. Parker, G. Bekey, and J. Barhen (eds.) *Distributed Autonomous Robotic Systems* Vol 4, Springer-Verlag, pp. 347–356, 2000.

197) M. Wooldridge and N.R Jennings. Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 10(2), pp. 115-152, 1995.

198) M. Wooldridge and N.R. Jennings. Cooperative Problem Solving. In *Journal of Logic and Computation, 9 (4)*, pp. 563-592, 1999.

199) S. Yamada and J. Saito. Adaptive Action Selection without Explicit Communication for Multi-robot Box-pushing. In *IEEE Transactions on Systems, Man and Cybernetics 31 (3),* pp. 398-404, 2001.

200) T. Yan and H. Garcia-Molina. SIFT – A Tool for Wide-Area Information Dissemination. In *Proceedings of USENIX Technical Conference*. pp. 177-186, 1995.

201) B. Yu and M. P. Singh, Searching Social Networks, In J.S. Rosenchein, T. Sandholm, M. Wooldridge and M. Yokoo (eds.) *Proceedings of the 2nd International Joint Conference on Autonomous and Multi-Agent Systems,* pp. 65-72, 2003.

202) S. T. Yu, M. G. Slack and D. P. Miller. A Streamlined Software Environment for Situated Skills. *In Proceedings of the AIAA/NASA Conference on Intelligent Robots in Field*, *Factory, Service and Space,* 1994.

203) J. Zamora, D. R. Millan and A. Murciano, Learning and Stabilising Behaviours in Multi-Agent Systems by Reciprocity, In *Biological Cybernetics 78*, pp. 197-205, 1998.

204) F. Zini. *Case LP. A Rapid Prototyping Environment for Agent Based Software*. PhD Thesis in Computer Science, University of Genoa, Italy 2001.

Appendix A : Derived Publications

This appendix lists all the papers that were published, or are currently being reviewed, that were derived from the work leading to this thesis.

1) D. Rodić and A. P Engelbrecht. Investigation into Applicability of Social Networks as a Task Allocation Tool for Multi-Robot Teams. *Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003)*, Singapore, Program and Abstracts, pp. 133, 2003.
2) D. Rodić and A. P Engelbrecht. Social Networks as a Coordination Technique for Multi-Robot Systems. *Intelligent Systems Design and Applications*, Springer, pp. 503-513, 2003.
3) D. Rodić and A. P Engelbrecht. Investigation of Low Cost Hybrid Three-Layer Robot Architecture. *Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003),* Singapore, Program and Abstracts, pp. 85, 2003.
4) D. Rodić and A. P. Engelbrecht. INDABA – Proposal for Intelligent Distributed Agent Based Architecture. *Computational Intelligence*, *Robotics and Autonomous Systems (CIRAS 2003)*, Singapore, Program and Abstracts, pg 85, 2003.
5) D. Rodić and A. P. Engelbrecht. Framework for Interaction in a Multi Agent Systems. *South African Institute of Computer Scientists and Information Technologists conference (SAICSIT 2002),* Port Elizabeth, South Africa, published on CD, 2002.
6) D. Rodić and A. P. Engelbrecht. Framework for Interaction in a Multi Agent Systems. *South African Institute of Computer Scientists and Information Technologists conference (SAICSIT 2003),* Proceedings of the Post Graduate Symposium, pp 30-32. Johannesburg, South Africa. 2003.
7) D. Rodić and A. P. Engelbrecht. Social Networks as a Task Allocation Tool for Multi-Robot Teams. *South African Computer Science Journal 33,* pp53-67, 2004.
8) D. Rodić and A. P. Engelbrecht. Interesting Features of Social Networks as Applied to Multi-Robot Teams Task Allocation. *Submitted to IEEE Transactions on Man, Machine and Cybernetics.* 2005

Appendix B : Acronyms

This appendix provides a brief summary of the most commonly used acronyms in this thesis.

ACL	Agent Communication Language.
API	Application Programming Interface.
AI	Artificial Intelligence.
BBR	Behaviour Based Robotics.
BSA	Behavioural Synthesis Architecture.
BDI	Belief-Desire-Intention architecture.
BLE	Broadcast of Local Eligibility.
CBSE	Component Based Software Engineering.
CNP	Contract Net Protocol.
DAI	Distributed Artificial Intelligence.
GPS	General Problem Solver.
INDABA	INtelligent Distributed Agent Based Architecture.
IT	Information Technology.
KQML	Knowledge Query and Manipulation Language.
KSE	Knowledge Sharing Effort.
MDP	Markov Decision Process.
MACTA	Multiple Automata for Complex Task Achievement.
MAS	Multi-Agent System.
NQC	Not Quite C.
OOP	Object Oriented Programming.
STRIPS	Stanford Research Institute Problem Solver.
XML	eXtended Markup Language.

Appendix C : Terms and Definitions

This appendix provides a brief summary of the most commonly used terms and definitions in this thesis.

Agent: A computer system, situated in some environment that is capable of flexible autonomous action in order to meet its design objectives.

Agency: A notion of characteristics that define an agent. In this thesis, the characteristics of agents are a*utonomy, interaction, collaboration and learning.*

Architecture: A general methodology for designing particular modular decomposition for particular tasks.

Auctioning Coordination Approach: An approach to coordination based on organisational sciences in general and in market based approaches in particular. It is widely used as a coordination tool in MASs.

Behaviour: An algorithm that acts as a control law that encapsulates sets of constraints in order to achieve a specific task.

Clique: A subset of agents that is defined by the existence of strong relationships between them.

Conflict: A negative interaction between agents in MAS.

Controller Layer: A layer in hybrid three layer architectures. The controller layer usually encapsulates behaviours that allow for fast, real-time, interaction with the environment. It is sub-symbolic in nature.

Cooperation: A process that promotes the optimal state of a MAS, by enabling positive interaction between agents in a MAS, usually requiring communication between agents.

Cooperative Problem Solving: A process that promotes cooperation between agents. The cooperative problem solving process consists of four sub processes, namely *potential recognition, team formation, plan formation* and *plan execution.*

Coordination: A process that promotes positive interaction and restricts negative interaction between agents in MASs. The most common coordination approaches have origins in biological or organisational sciences.

Deliberator Layer: A layer in hybrid three layer architectures. The deliberator layer usually reasons using symbolic reasoning techniques such as inference and backward chaining. The deliberator layer also maintains a symbolic world model.

Hybrid Architecture: An architecture that uses both symbolic and sub-symbolic knowledge representation and exploits the strengths of each approach.

Interaction Layer: A layer introduced in the INDABA framework. The main purpose of the interaction layer is to facilitate coordination between the agents in a MAS, by providing an easy way of encapsulating a coordination mechanism in the agent architecture.

Learning: The ability of a system to learn, based on previous experience from its interaction with the environments, by improving its performance.

Multi-Agent System: A society of agents.

Multi-Robot Team: A society of robots.

Reactive Architecture: An architecture that is used for implementing reactive agents. A reactive architecture is sub-symbolic in its nature. The central premise of reactive architectures is that intelligent behaviour will emerge from agent's interaction with its environment.

Robot: An agent embedded in a real physical body in a physical environment.

Sequencer Layer: A layer in hybrid three layer architectures. The sequencer layer usually serves as an interface between the deliberator layer (that uses a symbolic knowledge representation) and the controller layer (that uses a sub-symbolic knowledge representation).

Symbolic Architecture: Architecture that contains an explicitly represented, symbolic model of the world, and in which decisions are made via logical (or at least pseudo-logical) reasoning, based on pattern matching and symbolic manipulations.

Social Network: A social network is a set of agents and a distinct relationship among the agents.

Social Networks Based Approach: A novel approach to coordination in MASs, based on the use of identified social relationships in a MAS.

Social Relationship: Relationships that link agents to each other. The relationships can either be positively or negatively weighted, and are directed. Examples of social relationships used in this thesis are trust and kinship.

Three Layer Architecture: The predominant hybrid robot architecture. The three layer architectures consist of three layers, namely *controller*, *sequencer* and *deliberator* layers.

Appendix D : Definition of Symbols

This appendix lists the commonly used symbols found throughout this thesis.

$\boldsymbol{T_k}$	A task that needs to be allocated to a multi-robot team, defined by the *n*-tuple $(T_{k1}, T_{k2}, \ldots, T_{kn})$.
$\boldsymbol{T_{ki}}$,	i-th attribute of task T_k .
$\boldsymbol{A_x}$	An agent in a MAS, defined by the *m*-tuple $(A_{x1}, A_{x2}, \ldots, A_{xm})$
$\boldsymbol{A_{xi}}$,	i-th attribute of agent A_x .
$\boldsymbol{A_{lk}}$	An agent selected as a team leader.
$\boldsymbol{R_i(A_{lk}, A_x, T_k)}$	The i-th relationships between the team leader A_{lk} and agent A_x in relation to task T_k.
$\boldsymbol{F_{xk}(A_{lk}, A_x, T_k)}$	Scoring function for agent A_x in relation to team leader A_{lk} and to task T_k.
$\boldsymbol{t(R_1, R_2, T)}$	Trust relationship that quantifies the reliability of robot R_1 in relation to R_2, based on the historical performance related to task T that has involved both robots.
$\boldsymbol{d(R_1, R_2)}$	Kinship relationship that is defined as the similarity between robots R_1 and R_2.