TRACE - AN ADAPTIVE ORGANIZATIONAL POLICY FOR MULTI AGENT SYSTEMS

THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF

DOCTOR OF PHILOSOPHY

BY

S SHAHEEN FATIMA

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES
SCHOOL OF MATHEMATICS & COMPUTER / INFORMATION SCIENCES

UNIVERSITY OF HYDERABAD
HYDERABAD - 500046
INDIA

DECEMBER 1999

Certificate

This is to certify that the thesis work entitled TRACE - An Adaptive Organizational Policy for Multi Agent Systems, being submitted by S Shaheen Fatima in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy (Computer Science) of the University of Hyderabad is a record of bonafide work carried out by her under our supervision.

The matter embodied in this thesis has not been submitted for the award of any other research degree.

[illegible] 9 Dec'99

Dr G Uma
Supervisor
SQL Star International
Somajiguda, Hyderabad

[illegible] 9. Dec'99

Dr P R K Murhty
Supervisor
Dept. of Computer and Information Science
University of Hyderabad

[illegible] 9.12.99

Prof. Arun Agrawal
Head
Dept. of CIS
University of Hyderabad

[illegible]

Prof. K Vishwanath
Dean
School of MCIS
University of Hyderabad

Declaration

I, S Shaheen Fatima, hereby declare that the thesis work entitled **TRACE - An Adaptive Organizational Policy for Multi Agent Systems** is the bonafide work carried out by me, as per the Ph.D. ordinances of the university. The matter embodied in this thesis has not been submitted for the award of any other research degree to the best of my knowledge.

[illegible]

S Shaheen Fatima

Acknowledgements

Though it is difficult to enunciate clearly all the help and support extended by several people, this is a small attempt to acknowledge their direct or indirect contribution in making me complete my research.

1 express my sincere gratitude to my supervisors Dr G Uma and Dr P R K Murthy for their guidance and support. I thank them for providing me with valuable suggestions and necessary facilities.

1 thank Dr G Uma for her help, suggestions and encouragement throughout this research. She has been very kind to sit for long hours and advise me in spite of her busy schedule. I learnt a great deal from every discussion that I had with her. I also thank Dr P R K Murthy for all his cooperation and support, especially when the research was at a very crucial stage. He has facilitated the successful and fast submission of this thesis.

Dr T S Perraju deserves special mention for the interest and enthusiasm he has shown in my research, despite being overseas. He read chapters of this thesis and offered helpful comments and insights, and criticized me when I was vague.

1 thank Prof. Arun Agarwal, Head, Dept. of Computer Science, for his support an guidance. 1 would also like to thank the staff of AI lab and Computer Center for providing me with the necessary facilities.

Further, I express my thanks to Prof. K Vishwanath, Dean, School of MCIS, for providing me with the necessary facilities and encouragement.

I also extend my thanks to M J College for its cooperation throughout this research.

Finally, but in no way the least, I thank my parents and family members for their valued support and interest in my academic activities.

I kneel before the Almighty for his blessings and helping me reach this stage, and mother nature for all the inspiration.

Abstract

This thesis proposes an adaptive organizational policy, called TRACE (Task and Resource Allocation in a Computational Economy), for MAS that operate under time constraints and varying computational loads. We view the MAS as consisting of several problem-solving organizations where each organization is comprised of multiple agents that may be grouped into teams for specific problem solving. Problem solving requests arrive at each of these organizations. A request that arrives at an organization is solved cooperatively by agents within that organization and independently of the other organizations. As the rate of arrival of problem solving requests at each of these organizations varies with time, there could be a situation where some organizations may have surplus resources, while others have insufficient resources and thereby turn down problem solving requests. In order to minimize these lost requests, the allocation of resources (agents) to organizations is changed dynamically using the microeconomic approach. This reallocation of resources changes the number of agents in the organizations and their skills, and is intended to balance the demand for resources at each organization with its supply.

Following a layered approach, we divide the problem of developing an adaptive organizational policy into two broad sub-problems viz.

1. Allocation of tasks to agents within an organization through the task allocation protocol (TAP).

2. Allocation of resources (agents) to each of these organizations using the resource allocation protocol (RAP).

These two protocols complement each other. The TAP allocates tasks to agents within an organization, whenever they are requested, and the RAP periodically determines the resource needs of all the organizations and reallocates resources to them in accordance with their needs

We present the results of our simulation studies that show the effectiveness of our approach in allowing the MAS to exhibit high performance despite unanticipated

changes in the environment. This generic framework can be used for realizing soft real time applications, which require tasks like condition monitoring, fault detection, diagnosis and treatment to be performed continuously.

Contents

List of Figures

Chapter 1
Introduction

This chapter provides the background, motivation and context for this research. It describes the Distributed Artificial Intelligence (DAI) approach to building large software systems and introduces the three major sub-fields of DAI - distributed problem solving, multi-agent systems and parallel artificial intelligence. The reason why agents need to communicate and cooperate, and the basic means of achieving this during problem solving are put forward. The fundamental modes of cooperative social interaction are described and the notion of organizational policy, which is the focus of this research, is introduced.

1.1 Motivation for Research

Multi-agent system technology is rapidly passing the purely scientific stage and entering the market, where it is being used to solve real world problems in a range of industrial, commercial and medical applications The main areas in which multi-agent systems are being used include the following [97]

- Real-time monitoring and management of telecommunication networks, where agents are responsible, for call forwarding and signal switching and transmission

- Improving the flow of air traffic, where agents are responsible for interpreting data arising at different sensor stations

- Monitoring and diagnosing faults in process control applications like nuclear power plants, where agents detect and diagnose faults

- Electricity transportation management, where agents are involved in the diagnosis of disturbances in electricity transportation networks and their restoration.

- Information handling in information environments like the Internet, where multiple agents are responsible for information filtering and gathering.

- Management of patient's health in a surgical intensive care unit (SICU), where agents monitor the condition of patients for occurrence of abnormal conditions and respond quickly to critical events.

- Electronic commerce and electronic markets, where 'buyer' and 'seller' agents purchase and sell goods on behalf of their users.

This rapid proliferation of multi-agent system applications places increased demands on the multi-agent system builder. The multi-agent systems of today are expected to operate in increasingly complicated environments that are dynamic and unpredictable. Consider the example of patient monitoring in an SICU, where agents monitor patients' health for detecting abnormalities (unanticipated changes in temperature, blood pressure, pulse rate and other relevant parameters) and react to critical events in real time [10] As another example consider the information gathering application, where a group of agents together satisfy the information needs of a user [49,100] The type and frequency of requests in this application varies non-deterministically. This characteristic of uncertainty of the environment in which agents operate is common to all the above applications Consequently, it is becoming increasingly important for researchers to address the issue of adaptability of the multi-agent system to changing environmental conditions Moreover, as agents have bounded rationality [69], it is possible for the problem-solving load on the multi-agent system to exceed its processing limit In addition to this, new areas of expertise may need to be added, when necessary, and the system must be able to evolve rather than being built afresh each time.

These requirements pose the following challenges to the multi-agent system builder:

- Agents must be able to

  i) adapt to unpredictable changes in problem solving environment, for instance when new information becomes available, it may invalidate existing beliefs or goals and

  ii) focus on higher priority tasks.

- The multi-agent system must be able to

  i) adapt to changes in load by diverting resources where they are needed most and

  ii) add new agents for problem solving in an incremental manner

and thereby reorganize itself dynamically.

Attempts have been made to address some of these issues in isolation, but an integrated framework is not available We therefore felt the need for a comprehensive organizational policy that addresses all these requirements. Such a framework will reduce the amount of effort required for building multi-agent system, and allow it to operate in dynamic and unpredictable environments

1.2 Introduction to Distributed AI

Early research in Al [103] dealt with developing theories and techniques to study and understand the behavior and reasoning properties of a single intelligent agent, that solved problems with minimal help from, and interaction with other systems (computer or human) [27] As individual systems increase in size and complexity new problems and limitations were noted [21]:

1. Lack of Scalability: The complexity of an agent rises faster than the complexity of the domain.

2. Lack of Versatility: A complex application may require the combination of multiple problem-solving paradigms.

3. Non-Reusability: Several applications may have requirements for similar expertise, which is to be coded afresh in each new situation.

4. Brittleness: Single agent systems fail when presented with problems even slightly outside their limited range of expertise.

5. Inconsistency: As an agent's knowledge increases it becomes correspondingly more difficult to ensure that the knowledge remains consistent and valid.

A powerful strategy to overcome the limitations of stand-alone AI systems is to create problem-solving organizations. Individual systems are put into a society of systems so that they can draw upon a diverse collection of expertise and capabilities. This is similar to the way people overcome their individual limitations by grouping together into societies , which accomplish what the individual cannot The ability to flexibly team up and coordinate group activities towards achieving individual and collective goals is a hallmark of intelligence and is the focus of DAI

1.2.1 Defining DAI

DAI draws on ideas, concepts and results from many disciplines like AI, computer science, sociology, economics, organization and management science Its broad scope and multi-disciplinary nature make it difficult to precisely characterize DAI in a few words [47] However the definition that is adopted in this thesis is

Distributing and coordinating knowledge and actions in multiple agent environments [1].

Individual problem solving entities called *agents,* are grouped together to form *teams* or *communities.* Each agent is capable of a range of useful problem solving activities, has its own goals and objectives and can communicate with other agents.

Agents in a community usually have problem solving expertise, which are related, but distinct. Frequently expertise of different agents has to be combined to solve problems. Such joint work is needed because of the dependencies between agents' actions, the necessity to meet global constraints and the fact that often no single agent has sufficient competence to solve the entire problem. There are two main causes of such interdependence [108]:

- When problem partitioning yields components which cannot be solved in isolation.

  In speech recognition, for example, it is possible to segment an utterance and work on each segment in isolation, but the amount of progress, which can be made on each segment, is limited. Allowing the sharing of hypothesis is a far more effective approach [144].

- Even if sub-problems are solvable in isolation, it may be impossible to synthesize their results because the solutions are incompatible or because they violate global constraints

  For instance when constructing a house, many sub-problems are highly interdependent (ex determining the size and location of rooms, wiring, plumbing, etc) Each is solvable independently, but conflicts, which arise when the solutions are collected, are likely to be so severe that no amount of work can make them compatible It is also unlikely that any global constraint (ex. total cost must be less than X) would be satisfied In such cases, compatible solutions can be developed only by having interaction and agreements among the agents during problem solving [96]

This analysis provides an indication of the type of social interactions, which should take place between agents. In the former, agents should transmit information, which reduces ambiguity. In the latter case, they should exchange partial solutions, underlying assumptions or similar information that will ensure consistency of the sub-problem solutions and satisfaction of global constraints.

Even when multiple agents work independently (ex. inferencing in standard logic or execution of pure functional languages), information discovered by one agent can be of use to another and can make the two agents solve the problem more than twice as fast. This acceleration or combinatorial implosion [13] can be particularly useful in domains with large search spaces.

Depending on the number of problems being solved, their nature and the way in which they are distributed, two types of DAI systems can be identified viz. distributed problem solving and multi agent systems. A third sub area called parallel AI deals with more traditional issues related to concurrent languages and hardware.

1.2.2 Distributed Problem Solving (DPS)

DPS is a cooperative activity of a group of decentralized and loosely coupled knowledge sources (KS) [108] The KSs cooperate in the sense that no one of them has sufficient information to solve the entire problem Information must be shared to allow the group as a whole to produce an answer Decentralized means that both control and data are distributed, there is neither global control nor global data Loosely coupled, implies that individual knowledge sources spend most of their time in computation rather than communication. All the KSs cooperate to solve a single task. DPS is characterized by the three phases shown in Figure I I [112] In the first phase, the problem is decomposed into sub problems [43,44,74,98] The problem solvers then communicate and cooperate to solve individual sub problems Finally, individual sub-problem results are integrated to produce the overall solution.

The concerns of the DPS researchers are with developing frameworks for cooperative behavior between willing entities rather than forcing cooperation as a form of compromise between incompatible entities [23,42,108,142]. Distributed Hearsay-II [117], distributed air traffic control [115], and distributed vehicle monitoring test-bed (DVMT) [143,144,145] are examples of DPS systems.

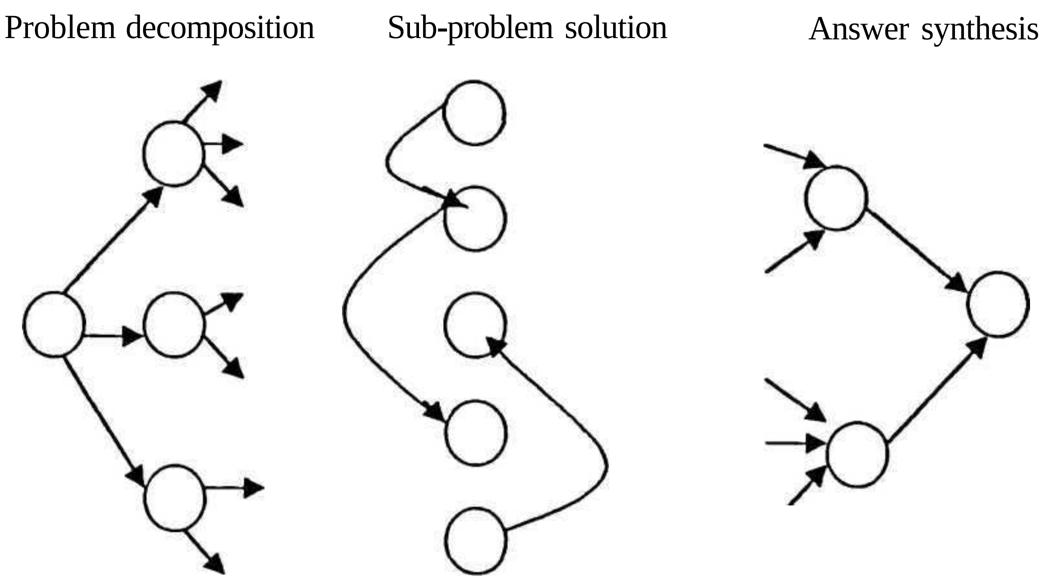


Figure 1.1 Phases of Distributed Problem Solving

1.2.3 Multi-Agent Systems (MAS)

Multi-agent system research is concerned with coordinating the knowledge, goals, skills and plans of autonomous intelligent agents so that they can jointly take actions or solve problems [1,71,97] Agents may work towards a single global goal or separate individual goals that interact in some way Like DPS systems, agents must share knowledge, data and results However, they must additionally reason about the process of coordination to ensure [96]

1. Coverage: all sub tasks of the problem solving activity must be included in the activities of at least one agent.

2. Connectivity: agents must interact in a manner, which permits the covering activities to be developed and integrated into an overall solution.

3. Capability: coverage and connectivity must be achievable within the network communication and computation resource limitations.

4. Coherence: team members must act in a coherent and consistent manner.

In general two paradigms are used to maintain coordinated behavior in dynamic and unpredictable environments:

- Centralized planning. A single intelligent agent constructs a plan to be carried cut by a group of agents and then hands out the pieces to the relevant individuals [60].

- Distributed planning. A group of intelligent agents together construct and possibly execute the plan [26,30,118]

The autonomy of agents makes the social interactions more complex Firstly agents must decide whether they wish to participate in a collaborative goal, hence the team organizer must send out requests for participation, rather than assume that other agents will always be willing to contribute In such systems the primary motivation for cooperation is self-interest the belief that participation in the social activity will be more beneficial than abstaining [31]

Secondly, it cannot be assumed that agents are benevolent' ready to change their goals to suit the needs of others and always be in agreement wit each other about the actions and their timings [59,92] Participants may already have activities planned for the time proposed by the organizer and may therefore be unwilling to drop them simply because they have been asked to do so. Agents only drop activities if they prefer the new proposal more than their current one [58].

In truly autonomous systems, the agent initiating the social interaction must exert sufficient influence over the others to persuade them to contribute, rather than assuming they will adopt goals upon recognition [17,50].

To summarize, traditional DPS research takes, as its starting point, that internal properties of the system can be assumed [28]. These properties include that agents will be truthful in their communications, that they will follow defined communication protocols, that they will promise to accomplish tasks when asked to and when they are able to and that they will perform tasks as promised [33]. Distributed problem solving systems assume these properties and are concerned with exploiting them to accomplish externally defined goals. On the other hand multi-agent system research is concerned with how to instill these properties assuming that agents are rational utility maximizers.

Agents in a multi-agent system possess the important characteristics of being *autonomous, intelligent* and *interacting* [22,47,63]

- Agents are *autonomous,* computational entities that can be viewed as perceiving their environment through sensors and acting upon their environment through effectors (see Figure 1 2) To say that agents are computational entities means that they physically exist in the form of programs that run on computing devices Autonomous means that agents have control over their behavior and can act without the intervention of humans Agents pursue goals or carry out tasks in order to meet their design objectives These tasks can be supplementary as well as conflicting

- *Intelligent* indicates that the agents pursue their goals and execute their tasks such that they optimize some given performance measures

- *Interacting* indicates that the agents may be affected, by the activity of other agents, in pursuing their goaJs and executing their tasks. Interaction can take place indirectly through the environment in which they are embedded (e.g., by observing one another or by carrying out an action that modifies the environmental state) or directly through a shared language (e.g., by providing information in which other agents are interested). Interaction is used to achieve coordination. The purpose of coordination is to achieve desirable states of affairs and avoid undesirable ones. To coordinate their goals, agents need to take dependencies among their activities into consideration. Two basic patterns of coordination are cooperation and competition [47]. In the case of cooperation, several agents work together and draw on the broad collection of their knowledge and capabilities to achieve a common goal. As against this, in the case of competition, several agents work against each other because their goals are conflicting [46]. When agents form a team and cooperate to accomplish a joint goal, they fail or succeed together In contrast to this, when agents compete, they try to maximize their own benefit at the expense of others, and so the success of one implies the failure of others.

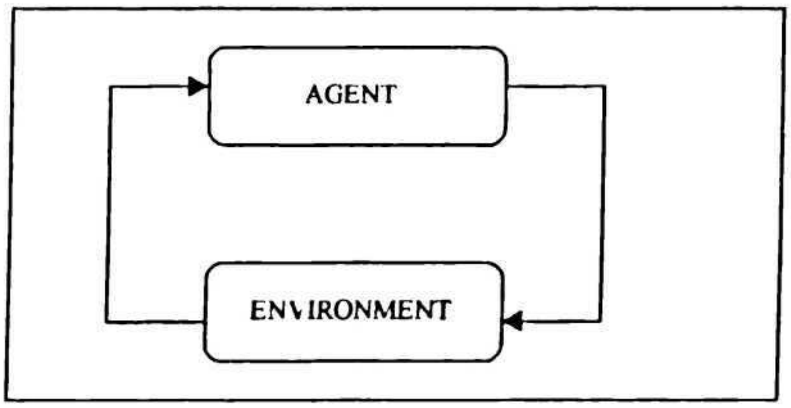


Figure 1 2 An agent in its environment The agent takes sensory input from the environment, and produces as output actions that affect it.

A multi agent approach to problem solving results in the following advantages [47]:

- Speed-up and Efficiency - Agents can operate asynchronously and in parallel, and this can result in an increased overall speed.
- Robustness and Reliability - The failure of one or a few agents does not necessarily make the overall system useless, because other agents already available in the system may take over their part.
- Scalability and flexibility - The system can be adapted to an increased problem size by adding new agents, and this does not necessarily effect the operation of the other agents.
- Costs - It may be much more cost effective than a centralized system, since it is composed of simple sub-systems of low unit cost
- Development and Reusability - Individual agents can be developed separately by experts, the overall problem can be tested and maintained more easily, and it may be possible to reconfigure and reuse agents in different application scenarios

1.2.4 Parallel AI (PAI)

PAI is concerned with developing parallel architectures, languages, and algorithms for AI. These are primarily directed toward solving the performance problem of AI systems and not toward conceptual advances in understanding the nature of reasoning and intelligent behavior among multiple agents [1]

PAI systems adapt to uncertainty in computation speeds, but not to alternative solution paths A DPS system is adaptive to uncertainty in problem-solving knowledge [140] but not to alternative problem contexts or to changing problem-solving roles for modules A multi-agent system is able to form and restructure coordination frameworks based on emerging contexts and changing problem-solving roles without the intervention of the programmer.

1.3 Means of Communication

Problem solving using the DAI approach involves communication among agents, as against single agent systems that only communicate with humans. The issue of communication is important as it could facilitate cooperative problem solving if addressed properly, or become a bottleneck otherwise. There are two major models for inter-agent communication [1]:

1.3.1 Shared Memory

The most widely used architecture is the blackboard system [9,110] The blackboard is a global database containing entries generated by the agents Entries are intermediate results generated during problem solving and include both, elements of problem solution,and important information for generating solutions Agents monitor portions of the blackboard waiting for particular patterns of data When such patterns occur, the agent takes the data and processes it, typically forming new combinations of data, which it places on another portion of the blackboard Distributed Hearsay-II [117] and DVMT [32,143] are examples of this approach

1.3.2 Message Passing

Message passing ideas are drawn from object based concurrent programming ACTORS [15,39,40,53] is a prime example of social interaction through message passing It does not require shared memory data structures between the communicating agents Actors are self-contained interactive, autonomous components of a computing system that communicate through primitives like create, send and become.

Message passing semantics are well understood, and offer a more abstract means of communication [13]. No hidden interactions occur, so there is greater comprehensibility, reliability and control over access rights. Unlike shared memory systems, message-passing architectures can be easily scaled up.

1.4 Forms of Social Interaction

Communication is the low-level description of how agents share information amongst themselves. It provides the means for social interaction. The term social interaction derives its use from the way people interact in a society to help each other. Several forms of social interaction are possible. This is also termed by various researchers as cooperation and coordination, and we use these terms interchangeably. The conditions for cooperation on a particular goal G to occur are [58]:

- The agents share a common goal.
- Agents are aware of the others' goals; ensuring cooperation is more than mere accidental coordination, as it incorporates an element of helpfulness.
- Agents must not only recognize other agent's goals but must have a preference for that goal.
- If attainment of the common goal requires attainment of sub-goals, then these are adopted on the same basis as above

The inclusion of context, G in this case, is needed to relativise the discussion. It is not usual to just say that two agents cooperate, rather they may cooperate on goal Gl, be independent of each other on goal G2 and G3, and be in conflict about G4

Two of the most fundamental forms of social interaction are [I]

task sharing and *result sharing.*

1.4.1 Task Sharing

Task sharing [112] is depicted in Figure 1.3. The numbers on the data flows represent the temporal ordering of events. In this form of interaction, one agent asks another to perform some problem solving activity for it. If the recipient accepts the request, it completes the task and informs the originating agent of the outcome. Task sharing has semantics similar to remote procedure call (RPC) in conventional distributed systems [41], although in some cases the requestor may be able to continuewithsubsequentactivity.

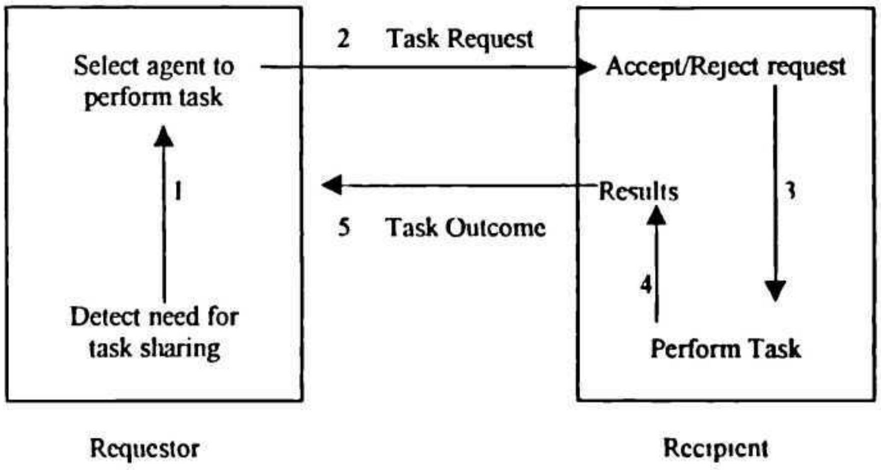


Figure 1 3 Task Sharing

A task sharing form of interaction may be invoked because the originating agent cannot perform the task itself, because it deems another agent m re capable or in order to balance the system's computational load A key concern for this form of social interaction is the mechanism by which agents decide who will perform which task.

1.4.2 Result Sharing

The second cooperation paradigm is that of result sharing, (see Figure 1.4). Agents assist one another by sharing spontaneously partial results, which are based on differing perspectives of a global problem. Different perspectives may arise because agents use different knowledge or different data (ex data that are sensed at different locations in the case of a distributed sensing system like in DVMT [31,143]). The agent who generates the information (in the form of partial results) evaluates whether it could be of use to any of the others. If it finds such an agent, the information is sent. When the recipient receives the information, it evaluates the information to determine whether it could be usefully incorporated into the current problem-solving context. If it can, then it is included; else it is discarded.

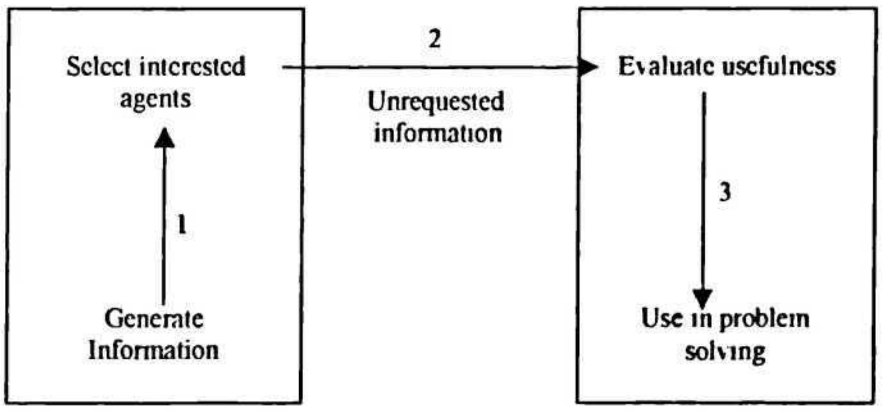


Figure 1 4 Result Sharing

Information sharing is most beneficial when sub-problems can not be solved by working independently, or when results achieved by the agent influence those produced by another The data may be used to confirm or deny competing results, to aggregate partial results, or to prune an agent's search space. It assumes the problem partitioning is carried out a priori, and that individual experts work on sub-problems which have some degree of commonality.

The functionally accurate cooperative (FA/C) distributed problem solving paradigm [140] is based on result sharing.

1.4.3 Joint Problem Solving

Task and result sharing occur in both distributed problem solving and multi-agent systems. Both have their limitations. Task sharing assumes that the requests can be solved by working independently. Result sharing does not incorporate a problem decomposition mechanism nor does it offer a feedback mechanism for coordinating the actions of groups of agents. Another type of social activity called joint problem solving, is effective in scenarios where the actions of several agents are intertwined [48,96,101,102]. Examples of joint activity include several agents lifting a heavy object, musicians in an orchestra, driving in a convoy, and playing cricket.

Joint action differs from both task and result sharing because it is a reciprocal process in which participating agents adapt their actions to comply with those of others. It is difficult to base the definition of joint action on external behavioral characteristics. The theories of joint intentions and shared plans [11], for joint activity are based on the assumption that the critical component is the mental state of the participants. A joint activity is one that is performed by individuals sharing specific mental properties, such as beliefs, desires and goals

1.5 Strategies for Cooperation

The power of distributed problem solving comes through effective cooperation and communication among agents If cooperation and communication are not effective then group problem solving performance may be worse than individual problem solving performance Considerable expertise is required to use communication effectively. Such expertise is referred to as cooperative strategies. Two distinct classes of such strategies are *organizational policies* and *information distribution policies [125]*.

1.5.1 Organizational Policies

Organizational policies dictate how a large task should be decomposed into smaller subtasks, which can be assigned to individual agents. Typically a given organizational policy assigns specific roles to each of the agents in a group. Such a policy is useful, if the resulting division of labor enables agents to work independently. For example, the corporate hierarchy is an organizational policy that is particularly effective if the corporate task can be decomposed in such a way that an agent at one level can work independently of others at that level, reporting results only to its immediate superior.

Organizational policies not only define task decomposition but prescribe communication paths among agents. They turn a random collection of agents into a network that is fixed at least for a given task. In the corporate hierarchy, the arcs between agents usually indicate which pairs are permitted to talk to one another, and in turn determine the nature of the messages that are allowed Such communication restrictions will be beneficial if they encourage only those agents that should communicate to do so; in particular agents that have dependent tasks or that may share resources. In general organizational policies strongly direct and constrain the behavior of distributed agents. They must be chosen appropriately for effective performance for a given task.

In distributed problem solving systems, groups begin by establishing an organizational policy To do so the agents must know which policy is appropriate to the current circumstances and implement it in a distributed fashion Any distributed method of implementing an organizational policy must answer the following questions:

- When does organizational structuring take place?
- How is the assignment of roles, specified by the policy, made to agents? In other words, how is the agent who is most appropriate for a given task found? This is known as the *task allocation* problem.
- When an agent is requested by another to conform to a role, or take another subtask for that agent, does the first agent have a right to negotiate? How does the agent weigh the value of competing tasks?

1.5.2 Information Distribution Policies

An information distribution policy addresses the nature of communication between cooperating agents. Decisions about how agents communicate with each other are constrained by the choice of organizational policies, since that policy decides the network of permissible communications. Within these constraints, a number of lower level decisions must be made about how and when communication should occur. Information distribution policy addresses the following issues

- Broadcast or selective communication. Are agents discriminating about who they talk to, if so what criteria are used to select recipients.
- Solicited or on-demand communication Assuming that agents know whom they want to communicate with, do they do so only if information is requested, or do they infer the informational needs of other agents and transmit data accordingly.
- Acknowledged or unacknowledged communication Do agents indicate that they have received information.
- Single transmission or repeated transmission communication Is a piece of information sent only once or can it be repeated? How frequently?

Poor decisions at this level result in the inefficient use of bandwidth and endanger global coherence by preventing agents whose tasks may interact from talking to one another. The goal of information distribution policies is to minimize these possibilities.

In contrast to this work which emphasizes the role of communication in cooperation, another approach has been developed to achieve cooperation in multi-agent systems without communication [92]. This is mainly intended for situations where communication is not available.

1.6 Objective of Research

The aim of our research is to develop an adaptive organizational policy that can deal with time constraints and computational load variations As we focus on soft real time domains, a major concern for us is to consider the criticality of tasks for their allocation. More importantly, there should be some means of preempting low priority tasks in preference to higher priority ones In addition to this, as the computational load is assumed to vary, we also need to provide some means of dynamically changing the skills of agents and their number in the multi-agent system This reorganization of the MAS should be done so that the demand for agents with a certain skill always matches the supply of agents having that skill.

Existing organizational policies address only some of these issues in isolation and hence cannot meet all the above requirements The objective of this research is therefore to develop a single comprehensive organizational *policy,* called TRACE (Task and Resource Allocation in a Computational Economy) that has the following characteristics.

1 Ensures coordination among agents when something unexpected happens.

2 Handles computational load variations and focuses res resources on the most critical tasks in the event of an overload.

3. Makes efficient use of resources.

4. Facilitates entry and exit of agents as problem solving requirements change over a period of time.

Our initial work on handling load variations and efficient use of resources are presented in [121,123].

1.7 Organization of the thesis

Chapter two provides a review of existing approaches for constructing intelligent agents. It describes the three types of agent architectures viz. deliberative, reactive and hybrid. The role of intentions in intelligent agents is highlighted and the limitations to using individual intentions to describe collaborative problem solving are discussed. The joint persistent goals formalism of Cohen and Levesque for collaborative problem solving and its extension, the joint responsibility model are explained in detail.

Chapter three provides a survey of the existing mechanisms for developing adaptive multi-agent systems for time constrained domains. The methods available for implementing organizational policies are explored and strategies that exist for dealing with time constraints and computational overloads are described The reason for the inapplicability of these methods to dynamic and unpredictable environments is also pointed out

Chapter four provides an overview of the proposed framework TRACE It describes the multi-agent system organization and individual agent architecture for TRACE The agent architecture is based on the joint responsibility model for collaborative problem solving The multi-agent system organization facilitates reorganization in response to changing environmental demands

Following a layered approach, we divided the problem of developing an adaptive organizational policy into two broad sub-problems viz. allocation of tasks to agents in an organization through the task allocation protocol (TAP) and allocation of resources to organizations through the resource allocation protocol (RAP). Chapter five explains the task allocation protocol.

Chapter six points out the limitations of existing methods that deal with computational overloads. It describes the resource allocation problem and highlights the advantages of taking an economic approach for resource allocation. The applicability of economic approach to our problem is shown and our resource allocation protocol (RAP) is explained. This chapter also presents the results of our simulation experiments that show the effectiveness of TRACE in meeting the objectives that were laid down.

Finally Chapter seven draws together the strands of research presented in this thesis and highlights some areas for future work. A brief account of the methodology used to guide this research is also given.

Chapter 2
Intelligent Agents: Approaches

This chapter introduces the three basic approaches, namely deliberative, reactive and hybrid, to building intelligent agents. The limitations of individual intentions for joint problem solving are listed and the joint persistent goals formalism of Cohen and Levesque [48,101,102], for collaborative problem solving is explained. This formalism forms the foundation for the joint responsibility model [96], which we adopt for our agents.

Section 2.1 introduces the three types of agent architectures Section 2 2 points out the role of intentions in an agent's reasoning process and explains the reasons for the unsuitability of individual intentions to joint problem solving Joint persistent goals, which is a formulation developed for joint problem solving is described in Section 2.3. Section 2.4 explains the joint responsibility model Finally Section 2 5 gives the conclusions.

2.1 Agent architectures

Based on the kind of representation and reasoning used, agent architectures are classified as deliberative, reactive and hybrid [97J

2.1.1 Deliberative architectures

Deliberative architecture is defined to be one that contains an explicitly represented symbolic model of the world (comprising mental states like beliefs desires and intentions - BDI) and in which decisions (for example about what actions to perform) are made via logical reasoning based on pattern matching and symbolic manipulation [3,6,94,126]

The reasoning process in a BDI agent is shown in Figure 2.1. As this Figure illustrates, there are seven main components to a BDI agent [47]

- A set of *beliefs* representing information the agent has about its current environment;
- A *belief revision function,* (brf), which takes a perceptual input and the agent's current beliefs, and on the basis of these, determines a new set of beliefs;
- An option generation function, (options), which determines the options available to the agent (its desires), on the basis of its current beliefs about its environment and its current intentions,
- A set *of current options,* representing possible courses of actions available to the agent;
- *Afilter* function (filter), which represents the agent's deliberation process, and determines the agent's intentions on the basis of its current beliefs, desires, and intentions,
- A set of current *intentions,* representing the agent's current focus - goals it has committed to trying to bring about,
- An *action selection function* (execute), \shich determines an action to perform on the basis of current intentions

A state of a BDI agent at any given moment is given by a triple (B D I) where BCBel, DCDes, and ICInt and which should a ways remain consistent

An agent's belief revision function is a mapping from the current percept and current beliefs to a new set of beliefs The option generation function, options maps a set of beliefs and a set of intentions to a set of deseres This function is responsible for the

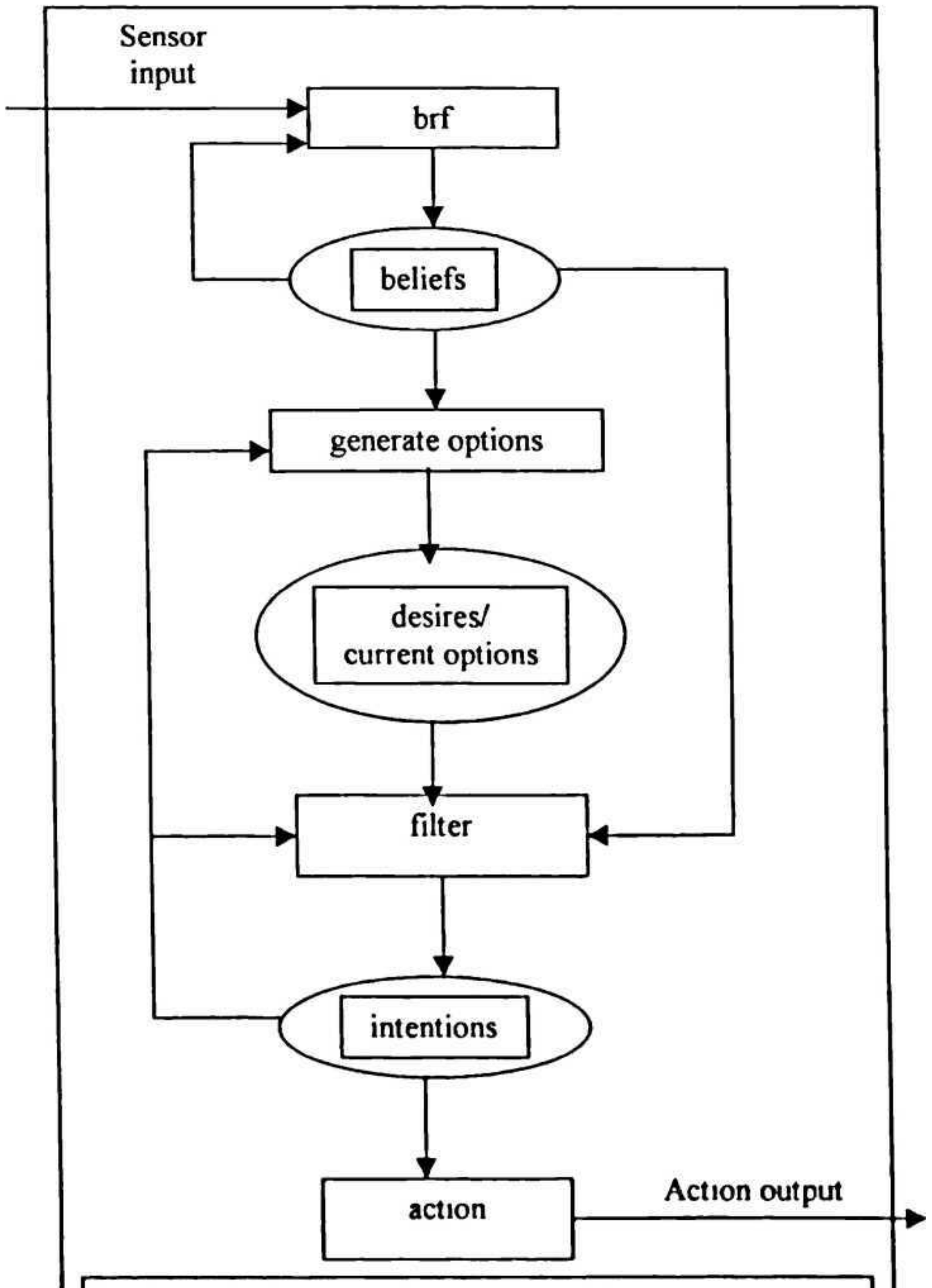


Figure 2 1 Belief - Desire – Intention Architecture agent's means ends reasoning - the process of deciding how to achieve intentions. Thus, once an agent has formed an intention to do *x,* it must subsequently consider options to achieve *x.* These options will be more concrete and less abstract than *x.* As some of these options become intentions themselves, they will also feed back into option generation, resulting in more concrete options being generated. The BDI agent's option generation process can therefore be thought of as one of recursively elaborating a hierarchical plan structure, considering progressively more specific intentions, until finally it reaches the intentions that correspond to executable actions. The options function should ensure that the options generated are consistent with both the agent's current beliefs and current intentions.

A BDI agent's deliberation process is represented in the filter function, which updates the agent's intentions on the basis of its previously held intentions and current beliefs and desires. This function does the following things' drops any intentions that are no longer achievable, retains intentions that are not achieved, and adopts new intentions.

The execute function returns executable intentions ones that correspond to directly executable actions.

The BDI agent's action function is defined by the following pseudo code

```
function action (p: P)          //P : percept
begin
        B:=brf(B,p)
        D:= options(D, I)
        I  =filter (B, D, I)
        Return execute (I)
end function action
```

The main advantages of BDI model are that firstly it is intuitive it is similar to the way humans go through the process of deciding what to do and then how to do it, and it is easier to understand Secondly, it gives a clear functional decomposition, which indicates what sorts of subsystems might be required to build an agent

However, providing an accurate and complete description and using it to reason in time are two main challenges of deliberative systems.

Planning agents for instance, fall under this category of deliberative systems. The best known early planning system was STRIPS [109]. The STRIPS planning algorithm was very simple, and proved to be ineffective on problems of even moderate complexity. Much effort was subsequently devoted to developing more effective techniques. Two major innovations were hierarchical and non-linear planning [34,35]. Another example is the Intelligent Resource Bounded Machine Architecture (IRMA), which is based on BDI model [82,83,84]

However, in real world things do not always go as planned. The assumptions of planning systems are that the environment is totally predictable and the internal model is totally complete and correct. These assumptions are often inappropriate - many environments are dynamic, ongoing, real-time and unpredictable To cope with this mismatch some researchers started investigating the idea of *reactive* systems [19,106].

2.1.2 Reactive Architectures

A reactive architecture is defined to be one that does not include any kind of central symbolic world model, and does not use complex symbolic reasoning [94] In such systems agents merely react to situations and do not reason about the world Usually, both, the agents and the actions are relatively simple and global properties are seen as emerging from the interaction of behaviors [113,114] The advantage of this approach is that because of their lack of explicit reasoning, agents are fast and can respond to changing environmental conditions so long as they have a predefined stimulusresponsepairing

Brooks [104,105,106] proposed the *sitbsumption architecture,* which is the best-known reactive architecture and built some robots using it to demonstrate his claim that intelligence does not require explicit symbolic representation and reasoning

There are two important characteristics of the subsumption architecture. The first is that agent's decision making is realized through a set of *task accomplishing behaviors*. Each behavior is an individual action function, which continuously takes perceptual input and maps it into an action. Each behavior is intended to achieve some particular task. These behaviors are implemented as rules of the form

Situation -> action

which map perceptual input directly to actions without the use of any symbolic representations or reasoning.

The second characteristic of subsumption architecture is that many behaviors can fire simultaneously. In order to choose between different actions, Brooks proposed arranging the modules into a *subsumption hierarchy,* with the behaviors arranged into *layers.* Lower layers in the hierarchy are able to inhibit higher layers the lower the layer is, the higher its priority The higher layers represent more abstract behaviors. Similar kind of work has been reported by Steels, who described simulations of 'Mars Explorer' systems, containing a large number of subsumption architecture agents [81].

The advantages of reactive architectures are their simplicity, economy, and robustness against failure. But purely reactive architectures have some unsolved problems. For instance, no principled method exists for building such agents While effective agents can be generated with small numbers of behaviors, it is more difficult to build agents that contain many layers The interaction between the different behaviors becomes too complex to understand

Though deliberative and reactive approaches appear to be opposite to each other, intentions actually provide a link between them [96] Reactive agents merely exhibit a special type of intention Each individual has a number of fixed, simple intentions, which are specified by the system designer and are implicitly available to the agent

When the designer defines an agent's behavior he is in fact installing its intentions. These fixed intentions (or precompiled behaviors) are then invoked automatically whenever certain conditions prevail; there is no run-time means-end reasoning. This contrasts with deliberative (non-precompiled) intentions, which are subject to means -end analysis. Jennings thus takes the unifying perspective that deliberative and reactive systems are just opposite ends of a spectrum rather than fundamentally different technologies. Similar reasoning is carried out by both types of system, although at different stages of the development process - run time for reflective systems and design time for reactive ones.

2.1.3 Hybrid Architectures

In most applications neither a completely deliberative nor a completely reactive approach is suitable for building agents. In such cases *tybrid* systems, which attempt to combine deliberative and reactive approaches are used, as is done in the Procedural Reasoning System [85].

One approach to build a hybrid agent is to use a deliberative component, which develops the plans, and a reactive component capable of reacting to events in the external world. Often, the reactive component is given some kind of precedence over the deliberative one, so that it can provide a rapid response to important environmental events. This kind of structuring leads to the idea of a layered architecture, of which Touring Machines [52], INTERRAP [55,56], and CIRCA [54] are good examples In such architectures, an agent's control subsystems are arranged into a hierarchy, with higher layers dealing with information at increasing levels of abstraction.

Generally, there is a minimum of two layers, to deal with the reactive and proactive behaviors respectively Broadly speaking, two types of information and control flow within the layers can be identified [47]:

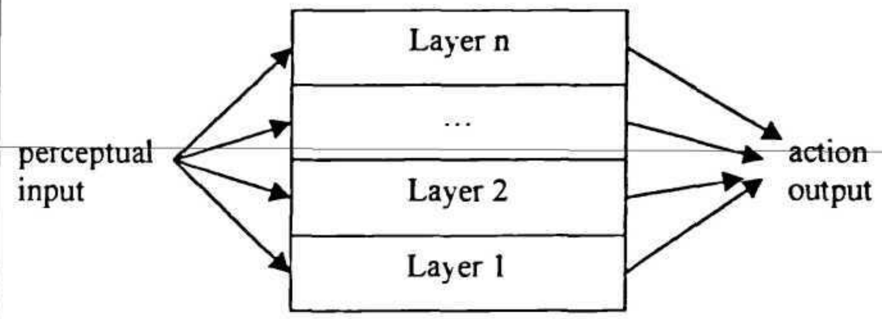


Figure 2 2 Horizontal layering

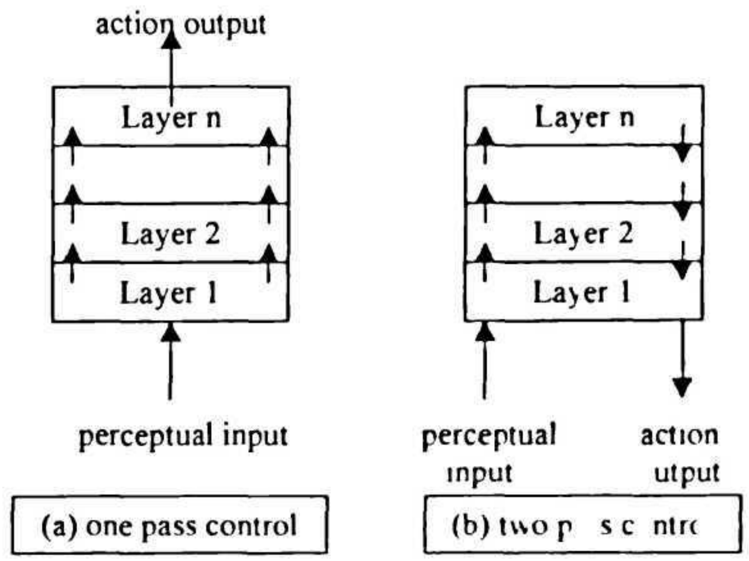


Figure 2 3 Vertical layering

- Horizontal layering

  In horizontally layered architectures, the software layers are each directly connected to the sensory input and action output. This is shown in Figure 2 2 [47]. Each layer itself acts like an agent, producing suggestions as to what actions to perform

- Vertical layering

  In vertically layered architectures, see Figure 2 3 [47], sensory input and action output are each dealt with by at most one layer each.

The advantage of horizontal layering is its conceptual simplicity if an agent is needed to exhibit *n* different types of behavior, then *n* different layers can be implemented. However, as the layers compete with one another to generate action suggestions, there is a danger that the overall behavior of the agent will not be coherent. To ensure consistency among layers some central control is required which makes decisions about which layer has control of the agent at any given time But the use of central control creates a bottleneck to the agent's decision making

This problem is partly overcome in the vertically layered architecture These are subdivided into one pass and two pass architectures In one-pass architectures, control flows sequentially through each layer, until the final layer generates action output In two pass architectures, information flows up and control then flows back down. Both these architectures reduce the complexity of interaction between layers

2.2 Intentions

From the above discussion it can be seen that intentions from an important component of intelligent agents. We therefore look at intentions in more detail Intentions mentioned in Section 2.1 are individual intentions. These define individual behavior and, as such, cannot be used for describing collaborative actions. There are two main reasons for this, [102]. Firstly, joint action is more than just the sum of individual actions, even if the actions happen to be coordinated (the coordination can be accidental). In order to represent this 'extra' part, formalisms and structures specifically related to collaborative activity are needed.

Secondly, there is a fundamental difference between individuals and groups. This can be shown by considering the notion of commitment. A group's commitment to an objective cannot be a version of individual commitment, where the team is taken to be an agent, because teams may diverge in their beliefs (this can only happen to schizophrenic individuals) [102]. If an individual comes to believe that a particular goal is unachievable, then it is rational for it to give it up The agent can drop the goal because it knows enough to do so Similarly, when an individual finds that the team's overall objective is impossible, the entire team must stop trying to achieve it. However, it is not necessary for the whole team to know enough to do so Consider the example of a group of agents collaboratively trying to lift a table One of the agents may observe that the table to be moved is nailed to the floor and therefore the group's objective cannot be attained However it cannot be assumed that all the other team members have also been able to make this observation and the corresponding deduction. Hence although there is no longer mutual belief that the goal is achievable, since one agent has seen the nails, there is not yet mutual belief that the goal is unachievable and therefore some parts of the team remain committed

Joint action requires an objective the group wishes to achieve and a recognition that they wish to achieve it in a collaborative manner [96] So in a cooperative lift, all team members must want to lift the table and they must want to do it as a team. The second component of this definition is important because it d simguishes between identical and parallel goals [107] For instance if *x* and y have the goal to have cake baked, their goals are identical, but if they both have the goal to eat cake (i e. *x* has the goal that *x* eats cake and y has the goal that *y* eats cake), they merely have parallel goals. This distinction is important because the two relationships imply different consequences in social interaction. Parallel goals give rise to competition if resources are scarce, whereas identical goals result in cooperation and coordination. We focus on identical goals, which are also termed common, joint or collaborative goals.

There are two prerequisites for collaborative problem solving to take place. A group of agents can engage in collaborative problem solving firstly by recognizing a shared objective and making a commitment to achieve it This shared objective binds the individual actions into a cohesive whole. However, having a common aim is not enough for attaining the objective in a collaborative manner. The next important prerequisite is that agents also need to agree upon a common solution and base their subsequent actions on it.

In order to develop a common solution, participating agents need to augment their individual intentions to comply with those of others So if one agent wants to lift a table at time 10 and another at time 15, then one or both of them will need to modify their intentions in order to make the joint action possible From the time a common solution is agreed upon till the completion of joint activity, all the agents in a team believe that the others are also doing their bit and adhering to the agreed solution [57]. Thus team problem solving is characterized by the mental state of the participants. A joint activity is performed by individuals sharing certain specific mental properties. Several social action formulations like the theories of collective intentionality [57], shared plans [11], we intentions [116], social plans [4], group intentions [86,95], joint persistent goals [48,101,102] and joint responsibility model [96], have addressed various aspects of mental state However, we adopt the joint responsibility model for agents in our framework because we are interested in dynamic and unpredictable environments and joint responsibility model ensures coordination among agents even when there are unanticipated changes in the environment The joint responsibility model is based on the work of Cohen and

Levesque on joint persistent goals. We therefore look at joint persistent goals in the following section and at the joint responsibility model in Section 2.4.

2.3 Joint Persistent Goals

Cohen and Levesque [101,102] propose a definition of joint intentions which is

Joint intention is a joint commitment to perform a collective act while in a certain shared mental state.

The central notion of joint commitment is formalized through the definition of joint persistent goals, which in turn is based on the concept of *achievement goals*. Achievement goals define the state of individuals participating in a team which is working on a collective goal (like move table) with a specified motivation (because it is necessary to gain access to a cupboard). Agent *A* has a *weak achievement goal,* relative to motivation q, to bring about *p* if either of the following is true:

- *A* does not yet believe that *p* is true and has *p* being eventually true as a goal (i.e. *A* has a *normal achievement goal* to bring about *p).*
- *A* believes that *p* is true, will never be true or is irrelevant *(q* is false), but has as a goal that the status of *p* be mutually believed by all team members.

Thus a weak achievement goal involves four separate cases either *A* has *p* as a normal achievement goal (it wants the table to be moved); thinks that *p* is true and wants to make this fact mutually believed (it believes the table has already been lifted); believes that *p* will never be true (it believes that the table is nailed to the floor) and wants to make this fact mutually believed or. finally, believes there is no longer a need to gain access to the cupboard *(q* is no longer true)

Weak achievement goals form the basis of the definition *of joint persistent goals* (JPGs). A team of agents has a JPG to achieve *p,* relative to 9. if and only if

1. they mutually believe that *p* is currently false (the table has not been lifted);

2. they believe that they all want *p* to be eventually true (they all want the table to be lifted); and finally

3. until they come to mutually believe either that *p* is true, that *p* will never be true or that *q* is false, they will continue to mutually believe that they each have p as a weak achievement goal relative to *q*.

Thus if a team is jointly committed to achieving *p,* the agents mutually believe that they each have *p as a* normal achievement goal initially. However, as time passes, team members cannot rely on the fact that they still have p as a normal achievement goal; they can only assume that they have it as a weak achievement goal. The reason for this is that one team member may have discovered privately that the goal is finished (true, impossible, or irrelevant) and be in the process of making this fact known to its associates.

If at some point, it is no longer mutually believed that everybody still has the normal achievement goal, then there is no longer a JPG as not all the agents wish *p* to be true In this case the team is no longer committed *to p.* However there is still mutual belief that a weak achievement goal will continue to persist which ensures that all team members are informed of the lack of commitment by an individual within the group.

A joint intention between agents *x and y to* achieve action a relative to motivation *q,* is then defined as a joint commitment to the agents* having done a collective action, with the agents of the primitive events as team members and with the team acting in a joint mental state.

This formulation provides criteria with which team members can evaluate their ongoing problem solving activity JPGs specify that agents must track a goal's validity and also provide the conditions, which must be monitored in order to perform this task. In addition to the evaluation criteria the model also provides the causal link to behavior; when an agent comes to privately believe that the joint act is untenable, or successfully completed, it must ensure that all its fellow team members are made aware of this fact.

JPGs however are not sufficient for attaining joint action firstly because joint action requires that agents work within the context of a commonly agreed solution, as there can be no joint action without this. A comprehensive formulation must make reference to plan states as well as goal states and explicitly specify the need for a common solution [96]. Joint action requires both a joint goal and adherence to a common solution while attaining it. Plan states should therefore be an integral component of the formulation of joint intentions. A code of conduct, which specifies how team members should behave with respect to the plan states, is therefore necessary. Secondly, an important requirement for being a team member is the ability to contribute something towards the group's objective. The joint responsibility model subsumes and extends the work on joint persistent goals on these two fronts. Responsibility uses joint intentions as *conduct controllers*, specifying how both individuals and collectives should behave while engaged in collaborative problem solving. The model addresses the problem of ensuring that groups remain coordinated in the face of unanticipated changes in the environment.

2.4 Joint Responsibility

In single agent systems, the behavior law is that of rationality: an agent only selects those actions that will lead to the satisfaction of one of its goals This kind of *individual rationality* is not sufficient for defining the behavior of participants engaged in cooperative problem solving. What is required is *team rationality*. The joint responsibility model proposed by Nick Jennings [96], for team rationality, has its roots in the theory of joint intentions of Cohen and Levesque [101,102], which was described in the previous section.

Responsibility extends the work on joint intentions by defining a structure for joint commitment, which involves both plan and goal states. It provides prescriptions of behavior for agents engaged in collaborative problem solving. The responsibility model addresses two facets of collaborative action. It defines preconditions that must be satisfied before joint problem solving can commence and prescribes how team members should behave once it has started.

Before joint problem solving can commence, four conditions need to be satisfied [96]:

An agent must recognize the need for collaborative action to solve a particular problem. This agent must contact other agents, who it believes will be able to play an active problem-solving role, to see whether they wish to be involved in bringing about the collective aim. An important constraint when carrying out this task is to ensure that all members are able to contribute something to the group's efforts

Once a group, which shares a common purpose, has been formed, members must agree that they wish to achieve the common objective in a collaborative manner

In particular, this involves agreeing they need a common solution, which they will adhere to Agents must agree for the duration of the joint action that they will follow a code of conduct with respect to their activity Adherence to this code (the responsibility model) ensures that the group operates in a coordinated and efficient manner even if things do not go according to plan

The second aspect of the responsibility model is applicable once the above conditions have been met and the joint action is initiated An agent follows the agreed plan of action (honoring its commitments) while continuously monitoring its local activity and information received from other agents to ensure everything is progressing smoothly (tracking the rationality of us commitments)

As agents are situated in dynamic and unpredictable environments, events may occur which affect their commitment. For example, newly acquired information may invalidate previous assumptions or important events may require urgent attention and distract the agent from its agreed course of action. Therefore conditions for dropping commitments (both to the overall objective and the common solution for obtaining it) need to be enumerated and the agent needs to continuously monitor its activity to detect when these conditions occur. If such circumstances do arise, agents need to reassess their position. This may require rescheduling actions, or re-planning, or dropping the joint objective altogether if the motivation is no longer present.

When an individual becomes uncommitted, it cannot simply stop its own activity and ignore others. This is because the other group members also need not necessarily be able to detect the condition for dropping the commitment Therefore the agent that realizes that a joint goal can no longer be achieved, must inform all its fellow team members of this fact and also the reason for its change of commitment This ensures team members are kept informed of events, which affect their joint work, and so when things do go wrong, the amount of wasted resources can be minimized

As we are interested in time constrained domains, reducing the amount of wasted effort is one of our major concerns Hence the architecture we chose for our agents also uses intentions for representing collaborative problem solving The agent architecture in TRACE is described in detail in Chapter 4

2.5 Conclusions

This chapter introduced the three different types of agent architectures Intentions are an integral component of intelligent agents As individual intentions are inadequate for joint problem solving, some formulation that is specifically related to joint problem solving is necessary The work on JPGs was described as it forms the foundation for the joint responsibility model that we adopt for our framework We chose the responsibility model because our focus is on dynamic and unpredictable environments, and the responsibility model ensures coordination among agents in such an environment.

Chapter 3
Towards Adaptive Multi Agent Systems for Time Constrained Domains: A Survey

This chapter lists the important characteristics of multi-agent systems and provides a survey of work related to our research.

Section 3.1 explains the important characteristics of mufti-agent systems. Section 3.2 explores some of the existing mechanisms for implementing organizational policies. These methods emphasize the role of communication in achieving cooperation through organizational policies. But in situations where there is no means of communication, or where the communication is unreliable, an alternate means of cooperation is required. Cooperation without communication is outlined in Section 3.3. Section 3.4 presents a discussion of the important properties of these methods and points out the reasons for their inapplicability to unpredictable and time constrained environments. Some strategies that exist for dealing with time constraints and computational overloads in multi-agent systems are explained in Section 3.5. This is followed by the conclusions in Section 3.6.

3.1 Characteristics of Multi-agent systems

One of the main concerns of multi-agent system research is to coordinate intelligent behavior among a collection of autonomous intelligent agents The focus of this research is on developing mechanisms by means of which agents can be made to coordinate with each other for joint problem solving. Agents must be able to define their own goals and plans and perform complex interactions with other agents. They should be independent of any particular problem solving organization, and be able to define and change the organization as the problem solving activity evolves. This is possible only when agents are adaptive in nature. Most situations consist of a collection of agents with various skills including sensing, communication, planning and acting. Such groups possess the following characteristics [70,73,80]:

1. Openness: Agents in the system are not statically predefined, but dynamically enter and leave the organization. This necessitates mechanisms for locating agents. Locating an agent is a challenging task, especially in environments that include large numbers of agents and that have information sources, communication links, and/or agents that might be appearing or disappearing

.

2. Autonomy: Each problem solver has to make local control decisions about what actions to execute with the information it has at hand when the decision is made. Agents need to co-ordinate their actions to promote beneficial interactions and avoid harmful ones because individual decisions have global impacts.

3. Adaptability: Multi-agent systems are expected to operate in a constantly changing environment. These include a change in one or more of the following:

The agent's environment, actions of other agents, availability of resources or problem solving demand on the organization.

The ability of the multi-agent system to adapt to such complex and dynamic environments by altering the problem-solving behavior of individual agents increases problem solving coherence.

4. Limited knowledge and perspective. These limit the capacity of an individual agent. This property of agents is called bounded-rationality [69] Bounded rationality is overcome by developing organizations of agents

5. Shared limited resources There are often shared limited resources with which an agent can execute tasks This in turn leads to bounded rationality [69]

6. Differing agent capabilities Agents typically differ in their appropriateness for a given task. The appropriateness of an agent for a task is a function of how well an agent's skills match the expertise required to do the task, the extent to which its

limited knowledge is adequate for the task and the current processing resources of the agent.

This characterization leads to two main difficulties in the development of multi-agent systems [1]. First there are difficulties *of optimal task assignment.* As many agents are appropriate for a given subtask, suitable task assignment mechanisms for optimal utilization of the agents' resources are required. Secondly, *task coordination problems* arise because tasks assigned to agents may not be independent. For instance, the execution of one task may facilitate or hinder the execution of another task.

To sum up, the main challenge is to develop a group of agents that can take a problem, work on different parts of it and together produce a solution. During the course of problem solving it is necessary to ensure that actions of individual agents are not only locally acceptable but that they are interfaced correctly with the actions of other agents [134]. The solutions that individual agents produce must not only be reasonable with respect to the local context but must also be globally coherent and this global coherence must be achieved by local computation alone.

These difficulties can be overcome by choosing an effective organizational policy [125].

3.2 Organizational Policies

An organizational policy addresses the issues of task and resource allocation, which are problems of assigning responsibilities and resources for a sub-problem to a problem solver. On one extreme the designer may make all the task assignments in advance, thereby creating a non-adaptive problem solving organization. This approach is inflexible, particularly in environments with a high level of uncertainty A better alternative is to have an adaptive task allocation mechanism that assigns tasks in a dynamic and distributed way Individual agents decide what tasks they will take on and how they will cooperate to achieve a goal collectively. This results in proper load balancing and bottleneck avoidance.

Some of the existing formalisms for implementing organizational policies in multi-agent systems are hierarchical organization, contract nets, social reasoning mechanism, and the use of economic methods.

3.2.1 Hierarchical Organization

In a hierarchical organization, decision-making and control is concentrated in a single problem solver at each level in the hierarchy. Interaction is through vertical communication from superior to subordinate agent. The subordinate agents have no autonomy. It is the superior agents that exercise control over resources and decision-making [24,71,79]. Hierarchical organizations are therefore not suited for autonomousagentinteractions

3.2.2 Contract Net Protocol

Contract net protocol [16,111,112] achieves opportunistic and adaptive task allocation among a collection of problem solvers using a framework called *negotiation* based on task announcement, bids, and awarded contracts It offers symmetry in the transfer of control process by using an interactive *mutual selection* (negotiation) process where the caller evaluates potential respondents from its perspective (via the bid evaluation process) and the respondents evaluate the available tasks from their perspective (via the task evaluation process) The evaluation procedures are local to the agents doing the evaluation.

Two kinds of agents, *manager* and *contractor,* exist in contract net protocol Managers announce tasks to contractors, which undertake them. When the manager has a task, it announces it to all contractors within the system Each contractor selects from the announced tasks, the one that best matches its own capabilities and bids for it. Finally the manager chooses what it believes to be the most appropriate bid and awards the task to that agent There may exist multiple managers and contractors in the system simultaneously. A manager can make multiple announcements simultaneously while a contractor can bid for only one task at a time.

Contract net protocol is based on human activity metaphor. The most significant feature of contract net protocol is that both managers and contractors award or bid according their own standards. This mechanism is called negotiation or mutual selection. Cheng and Ishida [16] investigated the various effects of mutual selection mechanisms on manager and contractor utilities. Their analysis (based on queuing theory) produced the following conclusions.

As system load increases, manager utility decreases. This is because when the system load is low, that is, the number of subtasks issued for bidding is small, managers receive bid messages from many contractors as soon as each task is announced. As a result the manager has a wide selection of bidders to choose from, and the manager utility increases. On the other hand if system load is high, the number of contractors dealing with tasks will increase and bid messages correspondingly decrease. As a result the managers selection of bidders shrinks and the manager utility decreases.

From the contractors viewpoint, their freedom of selection is restricted at low system loads, and contractor utility decreases When the system load is high more tasks are broadcast, allowing the contractors more freedom in selecting from the broadcast task at will. As a result contractor utility increases Therefore high system loads favor the contractors while low loads favor the managers

In the initial versions of this protocol agents were assumed to provide their services without any motivation. Sandholm [132] describes a variant of the contract net protocol for e-commerce in which tasks can be clustered to allow individual agents to bid for complementary tasks as a bundle Further extensions were made by

Sandholm and Lesser [119,133] where decornmitments and decommitment penalties were introduced.

Other applications of contract net protocol include Enterprise [131], the transportation control system TRACONET [132] and factory floor operations [50].

3.2.3 Social Reasoning Mechanism (SRM)

This model is based on social power theory [17,18]. In the SRM model [61,62,64,65,66, 67] dynamic coalitions are formed on the basis of motivation in the form of social dependence relations. Coordination starts in a recognition stage. This stage is seen as the trigger of negotiation, in which agents assess the potential for cooperation by recognizing how they depend on each other.

In SRM, agents maintain models of acquaintances in a data structure called external description (ED). An ED contains the agent's goals, plans, actions, and resources.

Using this external description an agent infers its dependence relation with other agents. An agent is said to be autonomous for a given goal if it has a plan that achieves this goal, and it can perform all needed actions and has control over all needed resources. If an agent is not autonomous for a goal, it is said to depend on another agent. An agent depends on another agent if there is a plan that achieves this goal and at least one action/resource appearing in its plan belongs to some other agents' action/resource.

Two kinds of dependence relations are defined [62] *mutual dependence* (MD) and *reciprocal dependence* (RD) A mutual dependence is said to exist between two agents if they depend on each other for the same goal A reciprocal dependence exists between two agents if they depend on each other for different goals. Mutual dependence leads to cooperation while reciprocal dependence leads to social exchange. These dependence relations are inferred by agents from the contents of the external description (ED) structure

An agent locally/mutually believes a certain dependence if it uses exclusively its own plans/its plans and those of others in order to reach a conclusion. Using these definitions a taxonomy of dependence situations is built This notion relates two agents and a goal. Six different cases have been defined:

1. Independence(IND): Using its own plans an agent infers that it does not depend on another agent for a certain goal.

2. Locally believed mutual dependence (LBMD): Using exclusively its own plans, an agent infers that there is a mutual dependence between itself and another agent.

3. Mutually believed mutual dependence (MBMD): Using both its own plans and those it believes the other one has, an agent infers that there is a mutual dependence betweenthem.

4. Locally believed reciprocal dependence (LORD): Using exclusively its own plans, an agent infers that there is a reciprocal dependence between itself and another agent.

5. Mutually believed reciprocal dependence (MBRD) Using both its own plans and those it believes the other one has, an agent infers that there is a reciprocal dependencebetweenthem.

6. Unilateral dependence (LJD): Using its own plans an agent infers that it depends on another agent for a certain goal, but this latter agent does not depend on it for any of its goals.

These dependence situations are used for establishing some decision criteria to help an agent to choose its partners when it is not able to achieve a goal alone A partially ordered set is constructed by defining a relation < on DS. where

$$DS = \{IND,UD.LBRD,LBMD,MBRD,MBMD\}$$

is a set composed of dependence situations. This partial order is based on two complementary criteria:

The nature ofthe dependence: It is always better for an agent to choose as partner, an agent that has the same goal (MD), because in this case the other would reciprocate. A second best choice is to choose as partner an agent that has an RD, since in this case the agent can propose an exchange ('if you help me to achieve my goal, I'll help you to achieve yours'). A UD is the worst choice, since the agent has nothing to offer in return.

The locality ofdependence: It is always better for an agent to choose as partner an agent that has a mutually believed dependence (MBMD or MBRD), since in this case the agent would not need to convince the other that its plan is better, which would be necessary to do if there is a locally believed dependence (LBMD or LBRD).

The combination ofthese criteria establishes the following partial order:

IND <UD < LBRD < {LBMD.MBRD} < MBMD

According to this partial order the best choice of partners for an agent are those whose dependence situation is MBMD, followed by either a LBMD or a MBRD, then by a LBRD and finally by a UD The two dependence situations LBMD and MBRD are incomparable according to the combination of the above two criteria [62].

Alonso [36] also has taken a similar kind of approach to group problem solving. In his work the dependence relations are defined more completely He uses two kinds of social dependence relations, *needs and prefers* These are defined as follows

An agent X is said to socially depend on an agent Y with regard to a set of actions A useful for achieving a goal G if

G is a goal of X and there exists at least one sub-goal Gi of G such that X is not able to achieve but Y is then X needs Y, denoted as NEC (X,Y,A,G).

X is able to achieve every sub-goal but prefers to satisfy the goal by agreement rather than alone then X weak depends on Y or X prefers Y, denoted as WEAK (X,Y,A,G).

The notion of mutual (M) and reciprocal (R) relations are defined in the same way as in Sichmans model described above. On the basis of these definitions two situations arise:

Symmetric situations where both agents depend on each other in the same way (i.e. symmetric necessity - SNEC or symmetric weak dependence - SWEAK).

Asymmetric situations where one agent X needs another Y, but Y only prefers to interact with X In this case Y is said to have power over X.

The space of cooperation can be represented graphically as in Table34 [36]. Here vertical numbers 1, 2, 3, and 4 stand respectively for NEC(X,YAGi), NEC(X,Y,A,G_2X WEAK(X,Y,A,Gi), WEAK(X,Y,A,G_2). The horizontal numbers stand for the same with X and Y interchanged.

	1	2	3	4
1	MSNBC	RSNEC	$\text{MEOW}_{x,y}$	$\text{RPOW}_{y,x}$
2	RSNEC	MSNBC	$\text{RPOW}_{y,x}$	$\text{MPOW}_{y,x}$
3	$\text{MPOW}_{x,y}$	$\text{RPOW}_{x,y}$	MSWEAK	RSWEAK
4	$\text{RPOW}_{x,y}$	$\text{MPOW}_{x,y}$	RSWEAK	MSWEAK

Table 31: Space of Cooperation

Sichmans model considers only cases 1.1,1.2, 2.1 and 22 because it allows agents with identical or different goals to cooperate but only in necessity cases.

Apart from these, other kinds of organizational structures have been developed. If the number of agents in the multi-agent system is very large, then agents encapsulate a model of their capabilities and send it to a *match maker or yellow pages* middle agent. Such a match maker agent can then be used to form different organisational structures like uncoordinated teams, federations or bureaucratic functional units [45,78,87,130].

3.2.4 Economic Approach for Decentralized Resource Allocation

The similarity of task allocation problems of multi-agent systems and economics resulted in the application of economic principles to multi-agent systems [14,147]. In this approach agents are participants in a computational economy, interacting in the market to further their own interests [139]. System behavior is defined in standard economic terms of production, consumption, bidding, and exchange.

The idea is to solve a distributed resource allocation problem by formulating a computational economy and finding its competitive equilibrium. To formulate a problem as a computational economy, the activities of interest are cast in terms of production and consumption of goods and agents are designed to choose strategies for production and consumption based on their own capabilities and preferences and the going market prices.

Computational economies consist of two kinds of agents, producers and consumers [12,139]. Consumer agents are endowed with an initial quantity of goods and engage in trades so as to maximize their utility. Producer agents are associated with a technology, which specifies an ability to transform come goods into other goods. The sole objective of producers is to choose an activity within their technology so as to maximize profits. From the agents' perspective, the state 6f the world is described by the going prices, that is, the prices determine the maximizing behaviors.

The basic problem of such computational economies is to obtain equilibrium of supply and demand across all the goods. Since these computational economies are instances of general equilibrium systems, the analytical tools and results of general equilibrium theory are directly applicable. In particular, the resource allocation models from the field of mathematical economics are applied to resource allocation in distributed computer systems. This gives solutions that are Pareto optimal, i.e. there is no solution that makes some agent better of without making some other agent worse off. Two basic microeconomic approaches towards developing decentralized resource allocation mechanisms have been identified [68], *price directed* and *resource directed* approaches.

Price Directed Approach

In the price directed approach an initial allocation of resources is made and an arbitrary set of system wide initial resource prices is chosen Prices are then literatively changed to accommodate the demands for resources until the total demand for a resource exactly equals the amount available. At this point the resulting allocation of resources is said to be Pareto optimal [8].

Resource Directed Approach

In this approach, during each iteration each agent computes the marginal value of each resource it requires given its current allocation of resources That is it computes the partial derivative of its utility function (performance) with respect to the resource, evaluated at the current allocation level These marginal values are sent to other agents requiring use of this resource The allocation of the resource is then changed such that agents with an above average marginal utility receive more of this resource and agents with a below average marginal utility are allocated less of the resource [68]. An attractive feature of this process is that when analytic formulas are used to compute performance, successive iterations of the algorithm result in resource allocations of strictly increasing system wide utility

Areas where market oriented approach has been applied include task allocation in agent based digital library [29,91,147], multimedia network applications [51], and optimal file allocation [68].

3.3 Cooperation without Communication

In all the above formalisms, cooperation among agents is achieved by means of communication. Although communication is a powerful instrument for accommodating interaction, it is possible to have situations where communication between agents is impossible, for example, as a result of communication equipment failure. Genesereth, Ginsberg, and Rosenchein [92] developed a framework for cooperation without communication among agents. They however assume that sufficient sensory information is available for individual agents, to deduce at least partial information about each other's goals and rationality. For example, an autonomous land vehicle in a battlefield may perceive the actions of another autonomous land vehicle and use plan recognition techniques to deduce its destination or target, even in the absence of communication.

The essence of interaction is the dependence of one agent's utility on the actions of another. This dependence is characterized by defining the payoff for each agent *i* in an interaction *s* as a function p_i that maps every joint action into a real number that designates the resulting utility for i Assuming that M and N are the sets of possible moves for two agents respectively, p^s_i, is

$$p^s_i : M \times N \to R$$

The values of this function are represented in the form of payoff matrices Agent choices are condensed into single choices and the cross product of all agents' choices is a state with a known payoff to all parties This approach generally assumes that agents have common knowledge of those final payofffs, have unlimited computational power, and that they are able to generate the complete payoff matrix and can find the *equilibrium points*.

Two strategies S and S' are said to be in equilibrium if assuming that the opponent is using S', the best an agent can do is use S. An equilibrium point is an outcome resulting from two agents use of equilibrium strategies. For example, in the payoff matrix in Figure 3.1, the strategies where agent K chooses move *b* and agent J chooses move *d* are in equilibrium. In the same matrix, there is a second equilibrium point where K chooses *a* and J choose *c*.

		c	D
K	a	1 2	-1 -1
	b	-1 -1	2 1

Figure 3.1 Two equilibrium points

Contextual clues can be used by agents to resolve multiple equilibrium points The payoff matrix has however abstracted away this contextual information Thus even in a case where the payoff matrix could be generated and where the agents are assumed to have common knowledge of the payoffs and infinite computational power, there may still be competing equilibrium points with no way to decide among them

Kraus and Rosenchein [120] use the concept of focal points to obtain coordinated action among agents without the use of communication A focal point is a conspicuous point of agreement to which interacting agents gravitate The methods an agent uses to represent and reason about the world significantly effect the way it searches for focal points They demonstrate how their approach offers a better alternate to the payoff matrix representation for achieving coordination without communication

3.4 Discussion on Organization Policies

The contract negotiation process in CNP offers a useful degree of flexibility making it well suited to AI problems whose decomposition is not known a priori. It is also well suited to problems whose configuration is likely to change over time. On the negative side, the contract net protocol is communication intensive since the task needs to be described to all the agents of the system [122]. However the degree of communication can be reduced if agents maintain a model of the other agents capabilities and use focused addressing. Contract net protocol also does not allow preemption of tasks and is therefore not suited to time constrained applications.

The process of goal adoption in SRM is more complex but it is also more realistic. CNP is based on the assumption that agents are cooperative and benevolent towards each other. Negotiation is considered simply as the sharing out or delegation of tasks between such benevolent agents in a system. The manager cannot induce others to adopt its goals because agents do not maintain knowledge about the intentions and goals of other agents. In SRM, agents solve the problem of how to induce others to adopt their goals on the basis of knowledge about the intentions and goals of other agents that is maintained in the ED. However, in the case of open multi agent systems, where agents enter and leave the system dynamically, the EDs may not be correct at all the times. A method for revising beliefs in such open multi-agent systems is proposed in [61.64.67J The presence of EDs facilitates focused addressing and results in reduced communication overhead However additional communication overhead is incurred during the exchange of information that is done to revise the EDs in an open multi-agent system

Further, the SRM does not quantify the dependence relations according to the importance of a given goal, which is an important consideration that needs to be made for real-time applications

In the micro economic approach, agents use profits as a means to induce other agents to adopt their goals. Task/resource allocation algorithms based on mathematical economics have the following attractive features They are simple and distributed in nature. Many ideas from economics can be directly applied instead of developing new theories. These methods converge to a solution rapidly and result in computation of successively better allocations at each step. Finally the solutions they produce are Pareto optimal.

3.5 Dealing with Time Constraints and Computational Overloads

Early AI systems addressed applications in which unlimited time was assumed to be available for problem solving. For many real applications the processing capability of AI systems may easily be exceeded by the processing requirements of the problem. In situations where the requirements change dynamically, AI systems must be able to rationally adjust their decision-making parameters to track changing problem requirements. To do this, new methods are being sought to provide accurate responses in the presence of time constraints One such method uses rational approximation [141] to make tradeoffs and compromises that favor timeliness over optimality. *A real-time agent is an agent whose utilityfunction depends on time.*

Agents in real-time environments need ways to control their deliberations They must be able to cease deliberation when action is demanded, and they must be able to use the available time for deliberation in order to execute the most profitable computations As AI systems move into more complex domains, all problems will become real-time because the agent will never have long enough to solve the decision problem because of the agent's bounded rationality [69] The agent is also subject to bounded reactivity because its sensors and actuators are limited in range, field of view and accuracy

In such situations an intelligent autonomous agent must interact with the other agents and the physical environment in real-time Because an agent cannot predict all of the events that will occur in the physical environment or result from other agents' reasoning, it must sense and respond to important unanticipated events. At the same time, because it has limited resources, the agent must be selective in its responses so as to achieve its most important goals. In general, the utility of an agent's behavior is a function of the criticality of the events to which it responds and the value of its responses to them [7]. Moreover, because other agents or physical processes in the environment have their own temporal dynamics, the *value* of an agent's response to an event depends not only on its response *quality* (the correctness of the response and other features such as completeness or precision), but also on its *response latency* (the delay between the occurrence and the response) Different situations may impose different constraints on response latency Deadlines deal with response latency. Deadlines are either soft deadlines, whose violation reduces response value incrementally, or hard deadlines, whose violation reduces response value directly to zero [10].

In addition to being individually challenging, these requirements, of meeting time constraints and solution quality, conflict making the construction of real-time agents difficult. Real-time AI systems must focus resources on the most critical events, and degrade gracefully.

What is therefore needed is a theory of algorithms that maximize the *comprehensive value* of computation [20] In other words, since the utility of a computation and the resulting action is a function of both, the quality of the resulting solution and the time taken to choose it, methods for designing algorithms that maximize this combined utility are needed Prominent among the techniques for construction of real-time agents is anytime algorithms [6]

3.5.1 Anytime algorithms and other approaches to build Real-Time Agents

Anytime algorithms is one of the most prominent approaches for designing real-time agents The term anytime algorithm was coined by Dean in the late 1980s in the context of his work on time dependent planning [129]. Anytime algorithms expand upon the traditional view of a computational procedure as they cover an entire spectrum of input output specifications, over the full range of runtimes, rather than just a single specification.

A standard algorithm is an implementation of a mapping from a set of inputs into a set of outputs. For each input that specifies a problem instance, there is a particular element in the output set that is considered the correct solution to be generated by the algorithm. An anytime algorithm is an implementation of a mapping from a set of inputs and time allocation into a set of outputs For each input there is a corresponding set of possible outputs, each of which is associated with a particular time allocation and some measure of its quality The advantage of this generalization is that computation can be interrupted at anytime and still produce results of a certain quality, hence the name *anytime algorithm* Figure 32 shows the performance profiles of standard and anytime algorithms A performance profile, which is a function that maps from the time given to an anytime algorithm (and in some cases input quality) to the value (quality) produced by that algorithm

Complex real-time systems are not built as one large anytime algorithm Systems are composed from components that are developed and tested separately However the optimal composition of two anytime algorithms (one of which feeds its output to the other) is not trivial Consider making a repair system from a 'diagnosis' component and a 'therapy' component The more time spent on diagnosis, the more likely the hypothesis is to be correct The more time spent on therapy planning, the more likely the problem is to be fixed, assuming the diagnosis is correct Zilberstien and Russel [124,128] describe one method for composing systems of anytime algorithms In this method, the user specifies the structural decomposition of a complex problem into elementary components, each of which is an anytime algorithm For example, a repair system might be specified as

```
    (defun repair (x)
```

apply-therapy x (diagnose (x)))

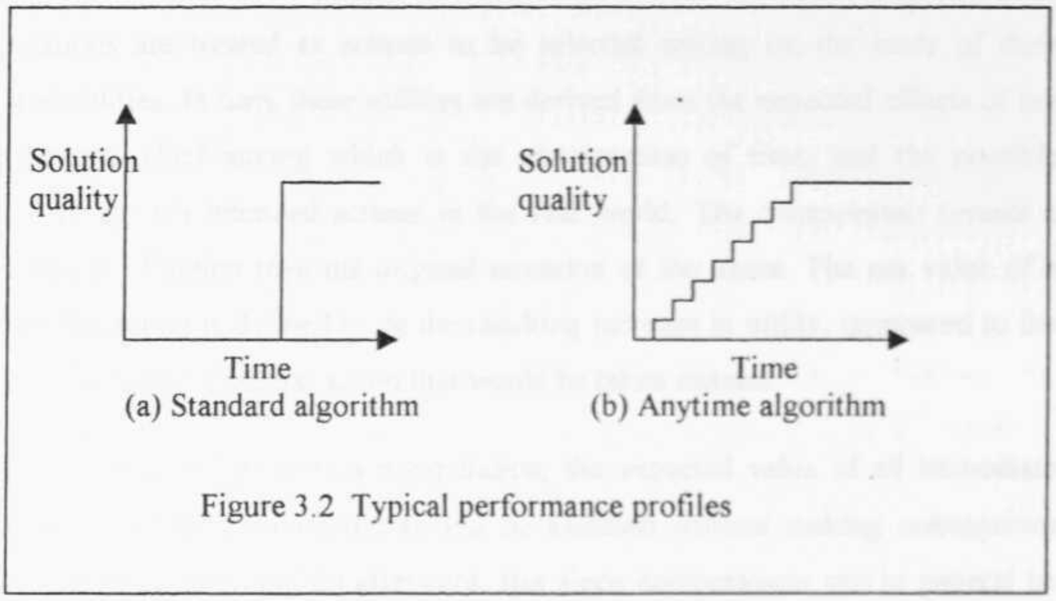


Figure 3.2 Typical performance profiles

Their method generates an anytime algorithm for the original problem by scheduling and monitoring the components in an optimal way with respect to the given utility function. A practical application of anytime algorithms is Guardian [7,10] that was developed for anytime diagnosis.

Other approaches for construction of real-time agents include approximate *processing* and *metareasoning* Approximate processing first elucidated by Lesser [141], is applicable in situations where satisficing answers are acceptable For thus to be successful it is necessary to have useful approximations with predictable effects for the application of interest, a problem solving architecture that allows these approximations to be represented reasonably, and control mechanisms for making decisions about which approximation to use Lesser's work on approximate processing [141] describes se\eral approximations for use in complex signal interpretation tasks These are categorized as *approximate search strategies, data* *approximations* and *knowledge approximations.* Decker applied the concept of approximate processing for a sensor interpretation application [75].

Russel and Wefald [127] introduced the idea of metareasoning, wherein computations are treated as actions to be selected among on the basis of their expected utilities. In turn, these utilities are derived from the expected effects of the computations, chief among which is the consumption of time, and the possible revision of agent's intended actions in the real world. The computation reveals a better course of action than the original intention of the agent The net value of a computation action is defined to be the resulting increase in utility, compared to the utility of the default external action that would be taken instead

Ideally, at any given point in a computation, the expected value of all immediate continuations of the computation should be assessed without making assumptions about what an agent would do afterward. But since computations can in general be arbitrarily long, such a complete analysis is infeasible One method proposed by Russel and Wefald [127] for this ideal case is the use of meta-greedy algorithms In this method the explicit consideration of all possible complete sequences of computation steps is avoided by considering *single* primitive steps and choosing the step that appears to have the highest immediate benefit Their experiments show that the resulting selection of computations is far better than random selection They also show that the resulting decision-theoretic meta-level exhibits anytime algorithm behavior.

3.5.2 Organization Self Design

Agent architectures discussed above attempt to meet deadlines by improving the decision making of individual agents Suitable structures to organize multiple real-time agents into useful problem solving entities are required Organization Self-Design (OSD) is an approach in this direction

Organization self design, proposed by Ishida et al, [136,137,138], is an approach for organizing real-time agents into multi-agent systems. Adaptive real-time performance is achieved through the reorganization of the society. Previous research on reorganization aimed at changing agent roles or inter-agent task ordering [62,111]. Organization self-design introduced new reorganization primitives, *composition* and *decomposition* of agents. It performed reorganization in real-time for problem-solving agents that use production system model Organization self-design is based on the concepts of organizational knowledge and reorganization primitives.

Organizational knowledge: To perform either domain problem solving or reorganization, agents need organizational knowledge, which represents necessary interactions among agents and their organization Organizational knowledge is formalized as a collection of *agent-agent relationships* and *agent-organization relationships*, which represent how agents' local decisions affect both other agents' decisions and the behavior of the entire organization

Reorganization primitives Organization self-design is performed through repeated application of reorganization primitives Two reorganization primitives, composition and decomposition of agents, dynamically change inter-agent relationships, the knowledge agents have about one another, the size of the agent population, and the resources allocated to each agent

The following situations may need reorganization

Change in the organizational performance level (e g, shorter or longer response time requirements or new quality levels)

Change in the level of demand for certain solution types (e g, more or fewer problem solving requests per unit time)

Changes in the level of demand for resources that the organization shares with others in its environment.

Under these circumstances no single organization can adequately handle all problems and environmental conditions. Problem solving requests issued from the environment arrive at the organization at variable rates. To respond, the organization must supply meaningful results within specified time limits, set by the environment and which may vary. These variations are changing conditions to which the organization must adapt using organizational knowledge arid OSD primitives

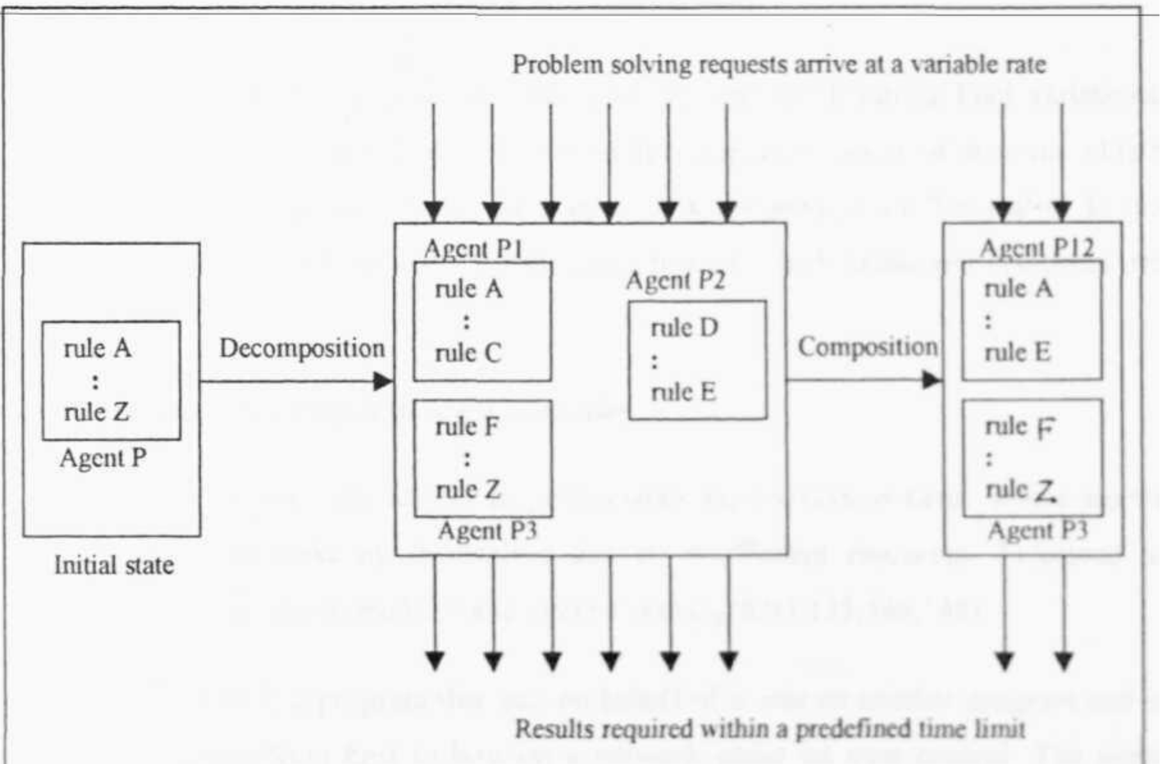


Figure 3.3 Composition and Decomposition

Figure 33 describes the process of OSD Composition and decomposition are repeatedly performed as follows

Decomposition divides one agent into two Decomposition is performed when the environment demands too much from the organization (eg . high arrival rates of problem solving requests), such that the organization finds it difficult to meet its response requirements with its available resources. Agents decompose to increase parallelism.

Composition combines two agents into one Composition is done for two reasons Firstly to free up computing resources when they are not required Secondly, agents compose to reduce response times. Composition may actually reduce response time, even though parallelism decreases, where coordination overhead (i.e., communication and synchronization overhead) is high

Although Ishida et al provide an interesting solution for handling load variations, their approach however does not address the important issue of how to obtain additional resources for creating new agents via decomposition They also do not consider the criticality of tasks for the execution of which additional resources are required.

3.5.3 Handling Computational Overloads

Multi-agent systems are subject to performance bottlenecks in cases where agents cannot perform tasks by themselves due to insufficient resources Solutions to overload include *agent mobility* and *agent cloning* [76,93,135,146,148]

A mobile agent is a program that acts on behalf of a user or another program and is able to migrate from host to host on a network under its own control 1 he agent chooses v\hen and where it will migrate, and may interrupt its own execution and continue elsewhere on the network The agent returns results and messages in an asynchronous fashion

The idea of self controlled program execution near the data source has been proposed as a better alternate to the client server paradigm [5] It offers a more efficient and flexible mode of communication (see Figure 3.4). The mobile agent paradigm has two general goals: reduction of network traffic and load balancing. It is currently being applied in the enhancement of telecommunication services [5,88]

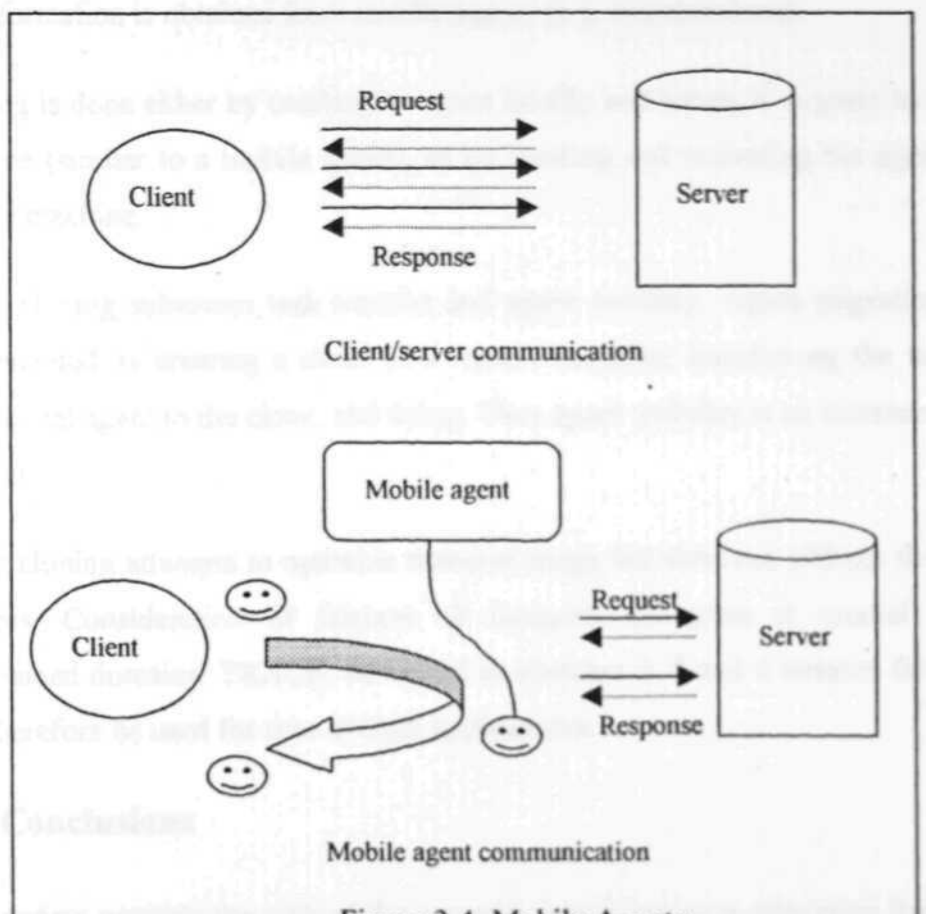


Figure 3.4 Mobile Agents

Agent cloning [76,99] is proposed as a more general approach to agent mobility Agent cloning is the act of creating and activating a clone agent to perform some of the agent's tasks for balancing the loads Cloning is performed when an agent predicts an overload, thus increasing the ability of the multi-agent system to perform tasks. To perform cloning, an agent must reason about its own load and loads of other machines. An agent considers cloning mainly if:

- It cannot perform all of its tasks by itself or by delegating to others

- There are sufficient resources for creating and activating a clone agent. This information is obtained from middle agents (e g matchmakers).

Cloning is done either by creating an agent locally and letting it migrate to a remote machine (similar to a mobile agent), or by creating and activating the agent on the remote machine.

Agent cloning subsumes task transfer and agent mobility Agent migration can be implemented by creating a clone on a remote machine, transferring the tasks from the original agent to the clone, and dying Thus agent mobility is an instance of agent cloning.

Agent cloning attempts to optimize resource usage but does not address the issue of fairness. Consideration of fairness of resource allocation is crucial for time constrained domains TRACE, described in chapters 4, 5 and 6 ensures fairness and can therefore be used for time critical applications

3.6 Conclusions

This review presents the state of the research in task, resource allocation frameworks, the methods for designing real-time agents, and methods for reorganizing the multi-agent system in response to computational overloads In order to obtain a truly adaptive organizational policy we require a complete and comprehensive framework that does the following

- Considers criticality of tasks,

- Allows individual agents to be adaptive, and also

- Allows the multi-agent system to dynamically reorganize itself.

As agents operate in environments where they neither have complete nor correct beliefs about the environment/ other agents, it is indeed essential for every agent to have the capability to engage in collaborative problem solving. However, as we are interested in situations with varying problem solving load, it is not enough if individual agents possess team rationality. There should also be a means of having the entire multi-agent system alter its organization in accordance with these variations and continue to provide services as per requirements, always giving preference to higher priority tasks in case of temporal conflicts A comprehensive framework that satisfies all the above requirements is not available TRACE framework described in Chapters 4, 5 and 6 strives to achieve the above properties.

Chapter 4
Organization and Agent Architecture in TRACE

Existing formalisms for implementing organizational policies assign specific roles to each agent in a multi-agent system. Examples are hierarchical organization, contract net protocol, social reasoning mechanism, and the use of matchmaker agents (described in Chapter 3). These policies allow the problem solving roles of the agents to change dynamically but do not adapt to variations in computational load on the multi-agent system. They are designed to operate for a predefined maximum problem-solving load and fail to respond, when the number of task requests, exceed thislimit.

On the other hand a decrease in the problem-solving load will result in surplus resources. What we need is a mechanism for dynamic allocation of resources to the MAS to allow it to operate in unpredictable environments Our objective is to find a suitable organization that exhibits high performance despite unanticipated changes in the environment (type and frequency of requests) and time constraints In such a scenario there is no single organization that yields optimal performance under all conditions. What is therefore required is some way of having the multi-agent system dynamically change its organization so as to always match its resources with the problem solving demand This requires changing the agents that comprise the organization and the organization structure Our approach is to have agents as flexible entities, which can be dynamically restructured in response to changes in the environment The knowledge that agents possess is also changed dynamically by migrating portions of it from agent to agent

In this chapter and the next two. we propose an adaptive organizational framework called TRACE It is assumed that agents obey the responsibility code of conduct for joint activity as proposed by Nick Jennings [96]. The resulting framework meets the needs of soft real time applications where the computational load cannot be predicted, and also utilizes system resources efficiently.

Section 4.1 gives an overview of the proposed framework. Section 4.2 describes the MAS organization for TRACE Agent architecture is described in Section 4.3. This is followed by the conclusions in Section 4.4

4.1 Overview

The proposed framework consists of several problem-solving organizations where each organization is comprised of multiple agents that may be grouped into teams for specific problem solving Problem solving requests with an associated priority and deadline arrive at the agents of these organizations A request that arrives at an organization is solved cooperatively by agents within that organization and independently of the other organizations The rate of arrival of problem solving requests at each of these organizations varies with time As a result, the requirement for resources also varies At any particular instant, some organizations may have surplus resources, while others have insufficient resources and thereby turn down problem solving requests In order to minimize these lost requests, the allocation of resources to organizations needs to be changed dynamically This reallocation results in reorganization of the multi-agent system and is intended to balance demand for resources at each organization with its supply ('resource' in this discussion refers to an agent).

The allocation of additional resources increases parallelism within the organization, resulting in an increase in the number of requests whose deadlines can be met However the total set of resources over all the organizations that constitute the multi-agent system always remains constant

This generic framework can be used for realizing soft real-time applications where the problem-solving load on an organization varies non-deterministically. For example, applications, which require tasks like condition monitoring, fault detection, and diagnosis to be performed continuously, can be implemented using TRACE.

Following a layered approach, we divide the problem of developing an adaptive organizational policy into two broad sub-problems viz.

1. Allocation of tasks to agents within an organization and
2. Allocation of resources (agents) to each of these organizations

This division simplifies the design of agents

The multi-agent system organization and the individual agent architecture for TRACE are described in this chapter The problems of task allocation within an organization and resource allocation to organizations are addressed in Chapters 5 and 6 respectively.

The multi-agent system organization that we propose facilitates reorganization to accommodate load variations and the agent architecture allows individual agents to exhibit team rationality and adapt to unpredictable changes in the environment

4.2 Multi Agent System Organization for **TRACE**

In TRACE, the multi-agent system is viewed as a collection of independent problem-solving organizations working under time constraints The elements that constitute an organization are

- The agents
- The organization structure
- Types of tasks the organizations carry out

Each of these is described below

4.2.1 Agents

Agents can make decisions and take actions, and are constrained by their organizational role. For example role may be that of a manager or a contractor. The actions of which the agents are capable depend on their capability and knowledge An agent's knowledge is comprised of task-based knowledge and organizational knowledge, i.e.. knowledge about other agents in the organization

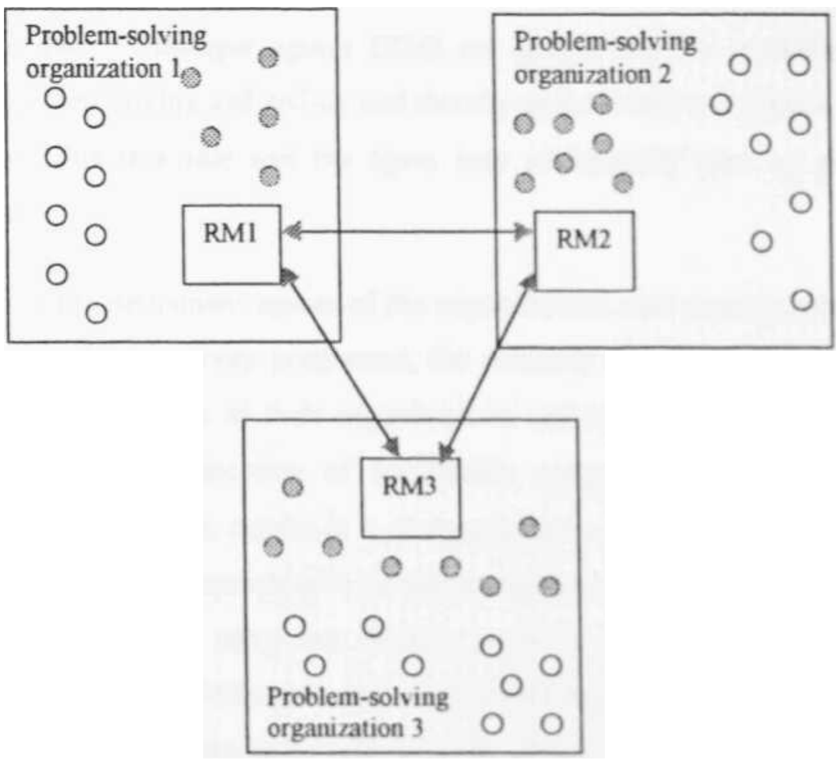

Problem-solving organization 1
Problem-solving organization 2
RM1
RM2
RM3
Problem-solving organization 3

Figure 4 1 Multi-agent system organization in 1 RACE

Agents are of three types

1 A fixed set *of permanent agents* (shown as blank circles in Figure 4 1) that an organization owns and that always belong to it. The number of agents in this set

is the minimum number of agents that are required to complete the organization and always keep it in operation.

2. A set of ***marketable agents*** (shown as shaded circles in Figure 4.1) which each organization can access. The agents in this set are dynamically allocated and are allowed to enter or leave any of the organizations The allocation of these agents is controlled by a special kind of agent called the *resource manager*. Every organization has an associated resource manager that keeps track of the resources required by it.

3. The ***Resource Manager agents*** (RM) set up markets for marketable agents and manage their buying and selling and thereby dynamically reorganize the multi-agent system. This is a role and the agent may additionally take up problem so ving activity.

Initially it is the permanent agents of the organizations that process incoming requests As problem-solving activity progresses, the resource managers periodically determine the current requirements of their organizations and use a market oriented protocol to arrive at a suitable allocation of marketable resources to the organizations This reallocation of resources results in a reorganization of the multi-agent system Every organization therefore consists of a set of permanent agents together with zero or more marketable agents that carry out domain problem solving activity The permanent agents play the role *of managers* and *contractors* but marketable agents only play the role of contractors These roles will become clearer when we describe the task and resource allocation protocols in Chapter 5 and 6 respectively

4.2.2 Organization Structure

The organization's authority and communication structure is described in terms of links among agents In the authority structure the links show who has authority over whom and thus reports to whom In the communication structure the links show who talks to whom. Figure 4.2 illustrates possible authority and communication structures. In a collaborative structure all links are possible, in a hierarchy there is a central or apex agent. As the agents that we consider are autonomous, TRACE communication and authority structures are collaborative.

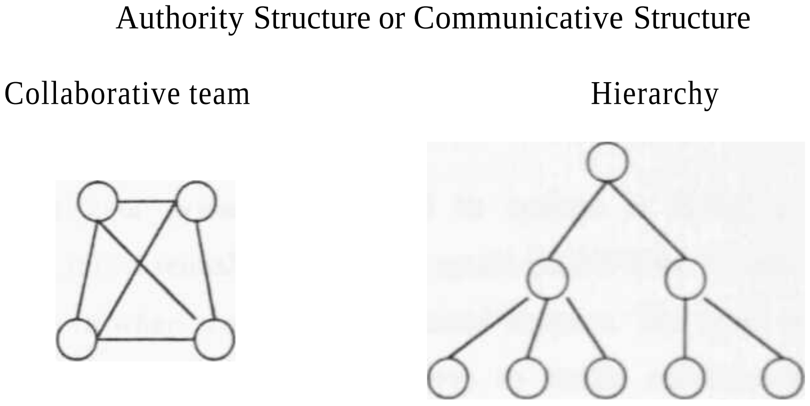


Figure 4 2 Organization Structures

4.2.3 Types of Goals

At any point of time the agents in an organization are engaged in executing goals A goal may be composed of sub-goals with dependencies among them Thompson [47] identified three such dependencies

- Pooled The results from two or more goals are jointly needed to perform a different goal
- Sequential Two or more sub-goals must be performed in a specified sequence
- Reciprocal Two goals depend jointly on each other

TRACE handles all three types of dependencies The set of goals faced by an organization can be thought of as its environment (or problem space) The type and frequency of requests (computational load) varies with time

Agents strive to achieve two types of goals: those which can be undertaken by individuals (primitive goals), and those in which groups (at least 2) agents work together (social goals). So for example, if two agents collaboratively lift a table, then the goal 'lift table' is social because it involves a team of agents Social goals ultimately give rise to primitive goals because only individuals have the ability to act. Thus social goal 'lift table' may give rise to primitive goals of agent 1 lifting at end 1 and agent2 lifting at end2. In the following discussion we use the term *task* or *action* to mean the sub-goals that are required to achieve a goal

As the multi-agent system is assumed to operate in complex and dynamic environments, it is essential to ensure that agents exhibit team rationality and remain coordinated even when something unexpected happens The agent architecture we propose therefore makes use of intentions to model collaborative multi-agent behavior.

4.3 Agent Architecture in TRACE

Figure 4.3 shows a high-level agent architecture for TRACH The rectangles correspond to processes and ovals to data stores This is a high level BDI architecture for collaborative multi-agent behavior in which intentions play a central role Intentions are used both to coordinate actions (future directed intentions) and to control the execution of current ones (present directed ones) An agent has a local knowledge base that includes the following

- ***Beliefs:*** Represent information the agent has about its current environment and are accessible to all the processes
- ***Desires:*** Represent possible courses of action available to an agent
- ***Intentions*** Represent the agent's current focus those goals thai it has committed to bring about

- ***Joint Intentions*** Represent the fact that a group of agents are jointly committed to a goal and the means for achieving the goal.

- ***Recipe Library.*** Contains goals and their associated recipes (plans) Recipes specify a set of steps (actions/tasks), with some temporal orderings that are necessary to accomplish a goal Tasks are also associated with a lower bound on time that indicates the minimum amount of time (for anytime solutions) that needs to be spent before its execution is terminated in order to get meaningful results.

- ***Organizational Knowledge*** Contains information about which other agents belong to the organization This knowledge changes dynamically as a result of reorganization.

Apart from this an agent has the following major functional components

- *Sensor and Communication Processor* Senses the environment and handles message traffic with other agents

- *Event Monitor* Checks for conditions that could result in dropping an existing intention and informs the override mechanism if they are satisfied Otherwise it generates a new objective and passes it on to the planner

- *Override Mechanism* Drops intentions that are decommited or for which motivation no longer exists When an agent receives information from its associates about decommitment or lack of motivation for a joint goal, the override mechanism drops the corresponding intention

- *Planner:* Takes the new objective and determines whether it can be met and if so how This is done on the basis of the recipe library and current intentions The output of this is the agent's desire

- *Inconsistency Revolver.* For local activities it checks whether the desire is consistent with the current intentions It then resolves inconsistency if any (on the basis of priority) and forms a new intention.

- *Contract Processor.* For social goals the contract processor finds suitable team members that can carry out the goals individual actions It then forms a joint intention for the goal after ensuring consistency of its actions with existing intentions.

- *Goal processor:* Takes individual intentions and executes them Intentions represent information about which task is to be executed and when The goal processor is assumed to have the knowledge required to execute individual tasks specified in an intention. In order to accommodate anytime algorithms, a monitor in the goal processor continuously keeps track of the time for starting the next goal. It terminates the execution of the present goal when the time to start next goal arrives Our protocol ensures that by this time the current goal has got its minimum chunk and has an acceptable result

Agents in an organization receive a stream of time-constrained problem solving requests from the environment and from other agents in the organization The objective of the event monitor process is to identify the following situations

- A new objective is raised

For instance, in a process control system, an agent may detect a fault and therefore should initiate the diagnosis process

- An event which is related to a local action occurs

An agent may be waiting for an acquaintance to provide a particular piece of information, when this event occurs the agent can continue with its processing

- An existing commitment is overriden

Commitments are not irrevocable, therefore an agent' must detect events which invalidate commitments so that it does not pursue fruitless activities

Assume that an event, which signifies the need for fresh activity, is detected by the *event monitor* process. This new objective serves as input to the *planner* process, which determines whether it should be met and if so how When deciding whether to adopt a new objective, the agent must consider its library of recipes and its current intentions. This allows the agent to determine whether the objective can be satisfied locally, or whether it necessitates social activity. Existing intentions must be taken into consideration while doing this, because they reflect activities the agent has already committed to. The output of the *planner* is a desire to pursue the objective locally, or to pursue it in a collaborative (social) fashion

If the desire is to pursue the objective locally, it must ensure that the new intention is compatible with the existing ones. Compatibility means that it does not conflict with anything the agent has already committed itself to For instance, for an agent capable of working on one task at a time, the decision for performing task t2 from time 5 to time 10 is incompatible with an earlier intention to perform a different task t1 from time 8 to time 15. If there are no inconsistencies, the new goal is added to the list of individual intentions and the agent commits itself to performing it The goal processor then executes the tasks specified in these individual intentions

If the new intention conflicts with the existing intentions, the inconsistency must be resolved; either by modifying the existing commitments or by altering the new intention so that it is no longer in conflict An important consideration in such situations is the agent's preferences or desires If the new goal is less important (less desirable) than existing ones, then it should be the one which is modified, conversely, if it is more desirable then it is the existing one which should adapted As a result of this modification, the less desirable tasks may not be able to complete as per their time schedule Such goals are decommitted Whenever an agent decommits

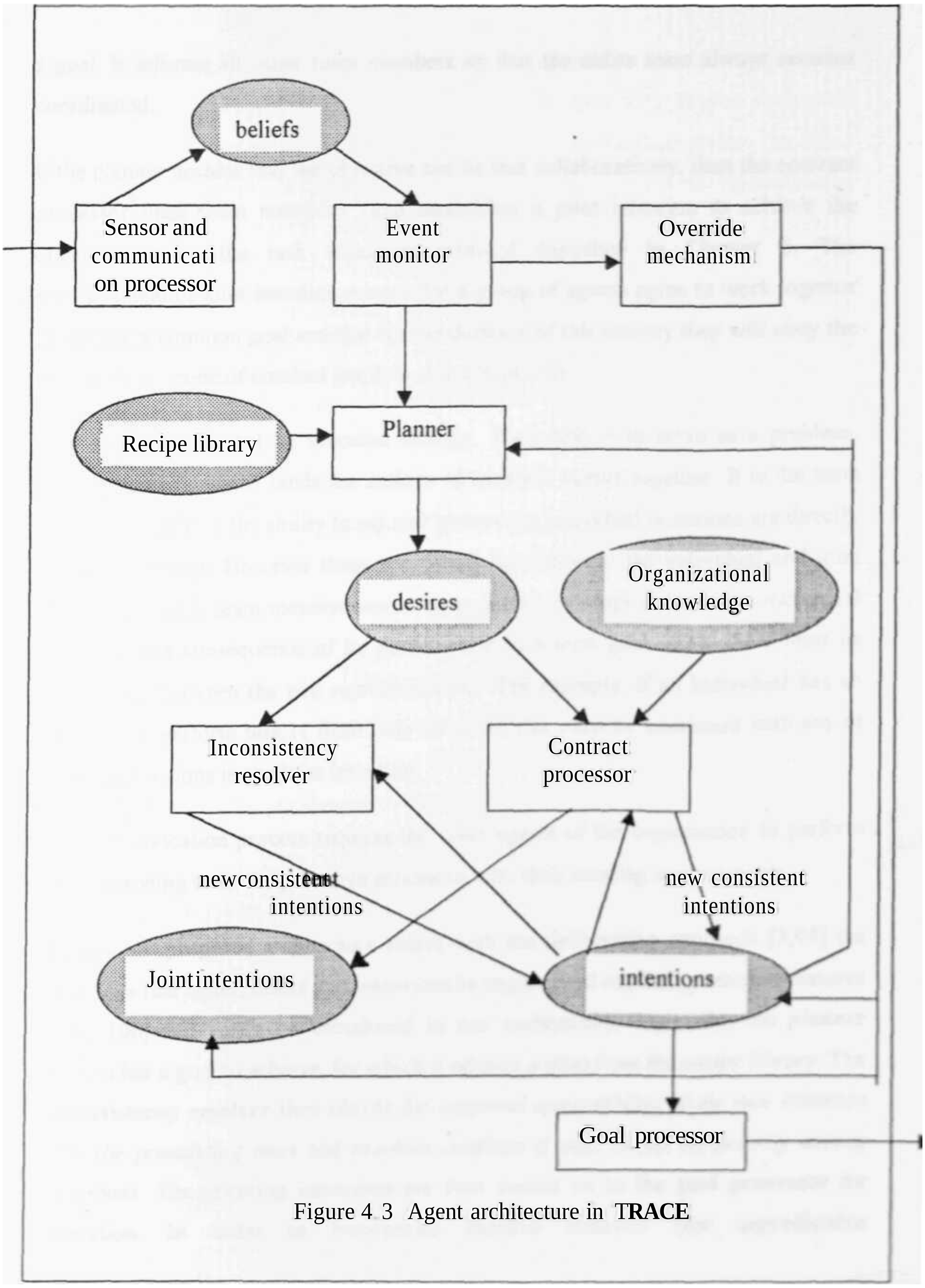


Figure 4 3 Agent architecture in TRACE

a goal, it informs all other team members so that the entire team always remains coordinated.

If the planner decides that the objective can be met collaboratively, then the contract processor finds team members and establishes a joint intention to achieve the objective, using the task allocation protocol described in Chapter 5 The establishment of joint intention means that a group of agents agree to work together to achieve a common goal and that for the duration of this activity they will obey the responsibility code of conduct (explained in Chapter 2)

Joint intentions cannot be executed directly Their role is to serve as a problem-solving context, which binds the actions of multiple agents together It is the team members who have the ability to act and hence only individual intentions are directly related to actions. However there is a causal link between the individual and joint intentions - each team member would be expected to adopt at least one individual intention as a consequence of its participation in a joint goal Also, there must be consistency between the two representations For example, if an individual has an intention to perform task tl from time 10 to 20, this must be consistent with any of the related actions in the joint intention

The task allocation process requires the other agents of the organization to perform local reasoning to fit the primitive actions in with their existing commitments

Though the proposed architecture shares with the deliberative approach [3,94] the basic idea that belief, desire and intentions be represented explicitly reactive features [105, 106] can easily be introduced in our architecture Normally the *planner* process has a goal to achieve, for which it *obtains a plan from the recipe library* The inconsistency resolver then *checks for temporal compatibility of the new intention with the preexisting ones* and *resolves conflicts if any, based on priority among intentions* The resulting intentions are then passed on to the *goal processor* for execution In order to incorporate reactive behavior (for unpredictable environments), high priority goals which require immediate attention need to be achieved quickly, without spending time on planning For this to happen such goals can be identified and an action for it stored in the **event monitor** When the **event monitor** encounters any of these goals, it passes the request on to the **inconsistency resolver** to accommodate the high priority goal Thus an intention can be generated for very critical activities without letting it go through the planner In this way new-intentions can prevail on preexisting ones and reactively modify the agent's course of actions. A hybrid approach is therefore attained, where agent behavior is reactive or deliberative depending on the actual situation

4.4 Conclusions

This chapter proposed an organization for the multi-agent system as well as the individual agent architecture. TRACE based applications are intended for use in unpredictable environments. TRACE agent architecture therefore allows agents to

i) Adapt to unpredictable changes in problem solving environment (by keeping its beliefs and goals always consistent with the latest information that it receives from the environment/ other agents)

ii) Exhibit team rationality by means of the joint intention representation and the override mechanism When the conditions for executing an intention no longer exist, the intention can be dropped through the override mechanism and fellow team members can be informed of this fact This aids in time constrained problem solving by preventing agents from pursuing fruitless activities and reducing the amount **of wasted** effort

iii) Focus on higher priority tasks. When an inconsistency is detected between an existing intention and a new one, it is resolved by the inconsistency resolver in favor of the higher priority intention.

In addition to this, the multi-agent system organization facilitates cooperative problem solving among agents within an organization, and allows the multi-agent system to:

i) adapt to changes in load by diverting resources where they are needed most,

ii) add new agents for problem solving in an incremental manner,

and thereby reorganize it dynamically The reorganization process is described in detail in Chapter 6.

Chapter 5
Task Allocation Protocol in TRACE

The organizational knowledge that agents in TRACE possess only specifies the list of agents that currently belong to its organization Agents need to identify team members for joint problem solving Since agents do not maintain explicit models of other agents' capabilities, a mechanism must be formulated for agents to find a 'capable' agent who is 'available' to perform this joint action This is the objective of the task allocation protocol (TAP).

As agents operate in complex and dynamic environments, it is necessary to ensure that the activities of agents always remain coordinated Joint intentions [101,102] guide problem-solving activity and play a key role in guaranteeing coordination among agents within an organization. In addition to this, we assume that agents possess anytime solutions [129] to goals. This is done so that an executing goal (that has anytime solution) can be terminated before its normal completion in order to accommodate higher priority requests

Section 5.1 gives an overview of the task allocation protocol Section 5 2 describes the task allocation protocol in detail The method for resolving temporal conflicts among intentions is explained in Section 5 3 Section 5 4 provides the results of our experiment and finally Section 55 gives the conclusions

5.1 Overview

Problem solving requests with priorities and deadlines arrive at each organisation The requests arriving at an organization are processed cooperatively by the agents of that organization and independently of other organizations

To establish joint activity, an agent must firstly recognize the need for it The agent who does this, is deemed the *manager* or organizer Each social action has one organizer and at least one team member called the *contractor* (an acquaintance who has agreed to participate). The manager's role involves obtaining a recipe, from the planner, contacting all the other agents of its organization to identify team members, determining when the actions will be performed and matching the team members with the actions to be performed.

If a goal can be achieved solely by the agent that receives it from the environment, it is a case of individual problem solving. Recognizing that a goal cannot be achieved all by itself puts the agent in an *organizer's* or *manager's* role It then has to seek *team members* or *contractors.*

Once the need for joint action has been ascertained, the responsibility model requires the following conditions to be fulfilled before it can commence, other agents who are willing to participate and are able to contribute something must be identified, the fact that a common solution is required needs to be acknowledged, participants must agree to obey the responsibility code of conduct (described in Chapter 2). and finally the common solution by which the social goal will be attained must be developed

The task allocation protocol does the following tasks

- Identification of team members in order to achieve a social goal, and
- Development of a common solution that is mutually acceptable to the organizer and the team members

As the organizer has a recipe, which specifies the sub-goals (tasks) and their temporal orderings, development of a common solution involves finding the actual time at which the sub-goals can be executed The fact that agents in a team will obey the responsibility code of conduct is implicit and docs not require agents to acknowledge this for every joint activity)

We aim at developing a task allocation protocol for open multi agent systems where agents dynamically enter and leave the system It is therefore difficult for agents to always maintain a correct model of others' capabilities In a scenario like this, ensuring that each of the above listed conditions is satisfied separately before the commencement of joint problem solving, involves a high degree of communication among agents. This will slow down the speed of operation of the system To overcome this difficulty, the protocol that we propose settles more than one condition in a single message interchange.

The process of finding a team member and agreeing upon a suitable time is done for every sub-goal of the recipe in the temporal order specified by the planner During this process, priorities are used to resolve any temporal conflicts that arise with preexisting commitments. The lower priority task is either rescheduled to accommodate a more critical task, or decommited altogether if deadlines make rescheduling impossible. Deadlines therefore ensure termination of the protocol

For the purpose of ensuring coordination among team members, all social activity is represented as a joint intention, which includes the list of team members, their roles and the common solution Whenever an agent reschedules or decommits a goal, it notifies all associated team members This keeps the entire team aware of the current state of problem solving activity and results in all team members either together progressing on the solution, or together dropping a goal if it is found unachievable

As our protocol is directed toward time constrained domains, the planner determines if an anytime solution is available for the sub-goals If so, it associates a minimum amount of time that needs to be spent for obtaining a meaningful solution After this period of time elapses, execution of the sub-goal can be terminated to accommodate more critical requests, or continued to completion otherwise

5.2 Task Allocation Protocol

The protocol is based on the following assumptions Firstly, it is assumed that the communication is foolproof and that the message delay time is known to all agents

Secondly, in order to carry out task allocation activity, agents share a global clock reference. Thirdly agents are able to accurately predict the time taken in terms of the global clock, to execute each domain level task. This facilitates the task allocation process and enables agents to make and honor commitments in a controllable manner.

The following notation is used in the discussion that follows

- a_i denotes action i
- A_i denotes agent i
- Gi denotes a goal
- Ti denotes the time at which action i is executed

After establishing that a goal is social, the organizer instantiates a representation of the social goal as a joint intention in its self-model (see Figure 5.1) The motivation slot indicates the reason for carrying out the joint intention The recipe is a series of actions, which need to be performed together with some temporal ordering constraints, which will produce the desired outcome The actions in Figure 5.1 (a_1, a_2, a_3, a_4) are temporally ordered The values l_1, l_2, l^3 and l_4 indicate the lower bounds on execution time for each of the actions, since actions are assumed to have an anytime solution. This protocol is however not limited to actions having anytime solutions. If an anytime solution is not available, then li is the fixed period of time that needs to be spent for obtaining the solution The recipe indicates what is to be done and in what order, not who is to do it nor the exact time at which it should be done

Problem solving requests are assumed to arrive with an associated deadline If an incoming request does not have an associated deadline, TRACE associates a default deadline with it This is done to ensure termination of the task allocation process

The start time and end time indicate the commonlv agreed time for starting and ending the joint activity The priority slot indicates the local agent's assessment of the importance of the intention and is used as the basis for computing its desuabilitv

Priorities are application dependent and the issue of determining priorities is therefore not addressed here.

The status slot of joint action description refers to the current activity of the task allocation protocol and has the value 'establishing group & developing solution' or 'executing joint goal'. Contribution slot records those agents that are capable of contributing and have agreed to the joint action. Initially the organizer, agent A_1, has agreed to contribute by performing the actions a_1 and a_3 No other agent has yet agreed or even been asked to contribute anything Contractors now need to be found for performing a^2 and 3_4.

Name: G
Motivation:
Recipe: a_1 1_1, a_2 l_2, $a_3 1_3$, a_4, 1_4
Deadline:
Start Time: End Time:
Priority: 23
Status: DEVELOPING SOLUTION / EXECUTING
Contributions:

A_1	ORGANIZER	a_1	t_1	AGREED
?	TEAM MEMBER	a_2	t_2	
A_1	ORGANIZER	a_3	t_3	AGREED
?	TEAM MEMBER	a_4	t_4	

Figure 5.1 Representation of Social Action (Joint Intention) in self model of A_1

Having identified the need for joint goal, the process of establishing it and arming at a common solution can commence. This requires finalizing the detailed timings and duration of the actions The team leader prepares an initial proposal for the individual action timings and fills in the joint goals duration and its start and end times in the joint intention

A1	a1	t1
?	a2	t2
A1	a3	t3
?	a4	t4

The timing proposal takes into account the fact that some time is required to agree to the solution, work cannot commence immediately. The formula for calculating the time lag is given below. It takes the following factors into consideration For each action which needs to be performed by an acquaintance at least two messages must be transferred (announcement to all agents of the organization and bid to the organizer) to establish its start time; an agent takes time to process a message and then an award message must be sent to all team members when the solution is agreed upon.

```
Start Time = current time +
             2 * number of nonlocal acts * communication delay -*-
             3*number of actions * estimated time to process message
             + communication delay
```

The team organizer does not have a complete picture of the capabilities of other agents within the organization. In particular the organizer does not know the existing commitments and desires of all its potential team members, so neither actions nor their exact timings can be dictated, they have to be negotiated To avoid chaotic behavior and many iterations, the organizer takes each action in the recipe in a temporally sequential order Note that only the task allocation is done sequentially - the tasks can be executed in parallel or overlapped if the plan has been so defined

Consider a case where $t_1=1^2$, $t_2=16$, t_3-21, $t_4=25$ For each action the organizer negotiates with the prospective team members (other agents of the organization) the appropriate time at which it should be performed. Thus action a_2 is negotiated first, and a time is agreed which fits in with the existing obligations of the prospective team member and the organizer's rating of the action's desirability (priority) Then A_1 finds a suitable time for a_3 and so on for each of the actions

As agents do not maintain models of other agents that represent their capabilities, the organizer describes the task to the entire organization (the list of agents that comprise the organization is maintained in the organizational knowledge mentioned in Chapter 4) by broadcasting a *task announcement* message This message, (see

Figure 5.2), indicates that the sender wishes to establish a joint goal and arrive at a common solution involving the recipient, states the team organizer's priority for the task, the action for which a contribution is required the time at which the action needs to be started, and a lower bound on execution time (for anytime solutions) that the prospective team member is expected to spend for that action.

Sender: A_1
Receiver: all agents within the organization
Type: TASK ANNOUNCEMENf MESSAGE
Contents:

Joint Goal	G
Priority	23
Contribution	a_2
Contribution Time	t2
Lower bound on time	1_2

Figure 5.2 Task Announcement Message

Upon receipt of proposal the team members evaluate it to see whether it is acceptable; refer to Section 5.3 for further details of this process If there is no conflict, the agent sets up a joint intention similar to that of Figure 5 1, and an individual intention as shown in Figure 5.3 The motivation slot indicates the goal for which contribution is required The status slot is 'pending'

Action: a_3
Motivation: Joint Goal G
Start Time: t_2

Duration:	Priority
Status Pending	Lower bound on time 1_2

Figure 5.3 Individual Intention Representation for Agent A_2

Agents then return a message indicating their acceptance to the team organizer (see Figure 5.4) The 'priority of decommited goal' slot indicates whether the prospective team member is able to accommodate the request bv decommiting a pre-existing lower priority goal, and if so, the priority of that goal The organizer can use this information as the basis for selecting a team member during the bid evaluation processs.

Sender: A_2
Receiver: A_1
Type: BID - ACCEPTANCE
Contents:
 Joint Action G
 Priority of decommited goal:
 Contribution a:
 Contribution Time: t_2
 Lower bound on time: 1_2
 Response: OK

Figure 5.4 Bid message indicating acceptance

If the suggested time is unacceptable, the prospective team member proposes a time at which the action can be fitted in with its existing commitments, makes a tentative commitment for this time and returns the suggestion to the organizer (see Figure 5.5). If the modified time is acceptable to the organizer, it will make appropriate adjustments to the subsequent solution timings and proceed with the next action If the modified time proposal is unacceptable, the organizer will look for a new agent to perform the action from its list of proposed contributors

Sender: A_2
Receiver: A_1
Type: BID - MODIFIED TIME
Contents
 Joint Goal G
 Priority of decommited goal
 Contribution a_2
 Proposed Contribution Time
 Modified Contribution Time t_2
 Lower bound on time
 Response NOTOK

Figure 5 5 Bid message with modified time

From among the agents willing to participate, the organizer selects as team member the agent that can perform the task earliest. If there is more than one agent that can perform the task, the organizer selects the one which can perform it by decommiting the lowest priority task.

The process of agreeing at a time for each action continues until all actions have been successfully dealt with. At this point the common solution is agreed upon and the organizer informs all the team members of the final solution by means of an award message (see Figure 5.6).

The joint intention status slot is changed to 'executing-joint-action' and the contribution slot is updated to indicate that all team members have agreed to the goal, and a common solution and implicitly to the responsibility code of conduct, and are now in the process of executing the joint action The status slot in the individual intention is also changed to 'executing' On receiving the award message, the team members also make similar changes to their joint and individual intentions and become contractors for that goal All the preliminaries for joint action have been satisfied and group action can begin

```
Sender: A1
Receiver: A2
Type: JOINT SOLUTION AGREED (AWARD)
Contents:
              Joint Goal G
              Solution
                     A1     a1     t1
                     A2     a2     t2
                     A3     a3     t3
                     A4     a4     t4
```

Figure 5 6 Notification of Start of Joint Action (Award Message)

After completing an allocated task, the team members report the results of execution to the organizer (see Figure 5 7)

Sender: A_2

Receiver: A_1
Type: RESULT OF EXECUTION
Contents:

Joint Goal	G
Priority:	
Contribution:	a:
Contribution Time:	t_2
Result:	
Time spent executing:	

Figure 5.7 Result of execution

5.3 Resolving Temporal Incompatibilities

In order to exhibit correct behavior, agents need to ensure that their intentions always remain compatible. Two intentions are said to be incompatible if the times for which they are scheduled overlap, they are compatible if they are distinct Consider an agent having two intentions for tasks a_1 and a: represented in its self-model as shown in Figure 5.8. These two intentions are compatible because the times at which they are carried out, 5 to 12 and 12 to 16 do not overlap

Name: a_1
Motivation: G_1
Start Time: 5 Max End Time: 12
Duration: 7 Priority: 10
Status Pending Lower bound on Time 7

Name: a:
Motivation G_2
Start Time 12 Max End Time 16
Duration 4 Priority 8
Status Pending Lower bound on Time 3

Figure 5 8 Consistent Intentions

Before the commencement of their execution a new request arrives that **corresponds** to the intention shown in Figure 5 9

Name: a_3
Motivation: G_3

Start Time: 15 Max End Time: 20
Duration: 5 Priority: 10
Status: Pending Lower bound on Time: 5

Figure 5.9 New Intention

The inconsistency resolver now has to determine whether the new proposal is compatible with the agents existing intentions. As a result of this analysis the inconsistency resolver will indicate that the new intention is compatible because even though the times overlap, a_2 requires an anytime solution (lower bound < duration) and can therefore be accommodated with the new intention. This requires termination of a_2 at time 15 in order to start a_3.

In case an anytime solution is not available for a_2, then it becomes incompatible with a_3. The inconsistency resolver resolves this by making use of the priority values for each of the intentions. If the new request is less desirable than the existing commitments, then the agent proposes a modified time that can be fitted in with the existing commitments. In the above example however, a_3 is more desirable Therefore the agent forms the intention to achieve a_3 from time 15 to 20 and reschedules a_2 after a_3. The new intention for a_2 now becomes

Name: a_2
Motivation: G_2
Start Time: 20 Max End Time: 24
Duration: 4 Priority: 8
Status: Pending Lower bound on Time 4

The other actions of G_2 that get affected due to this change also need to be rescheduled. If the new schedule for G_2 does not conform to its deadline, then the agent decommits G_2 and updates the number of recommitments This **is** all that needs to be done if G_2 is a primitive goal However if a_2 corresponds to a joint **goal** just rescheduling a_2 is not enough The agent must inform all team members about **its** decommitment to the originally agreed solution (see Figure 5 10)

Sender: A_2
Receiver: All team members
Type: DECOMMITMENT TO SOLUTION
Contents
Joint Goal G_2
Priority
Contribution a?
Old Contribution Time: t_2 New Contribution Time: t_2
Lower bound on time:

Figure 5.10 Decommitment to solution Message

Upon receipt of this message, the other team members also drop commitment to the common solution. When the team organizer receives this message it reschedules G_2 if possible, otherwise decommits the goal G^2, informs all team members about the decommitment to the joint goal (see Figure 5.11), and updates the number of decommitments. In this way all organizers record information about their decommited goals and convey this information to the resource manager, which utilizes it for performing resource allocation (explained in Chapter 6)

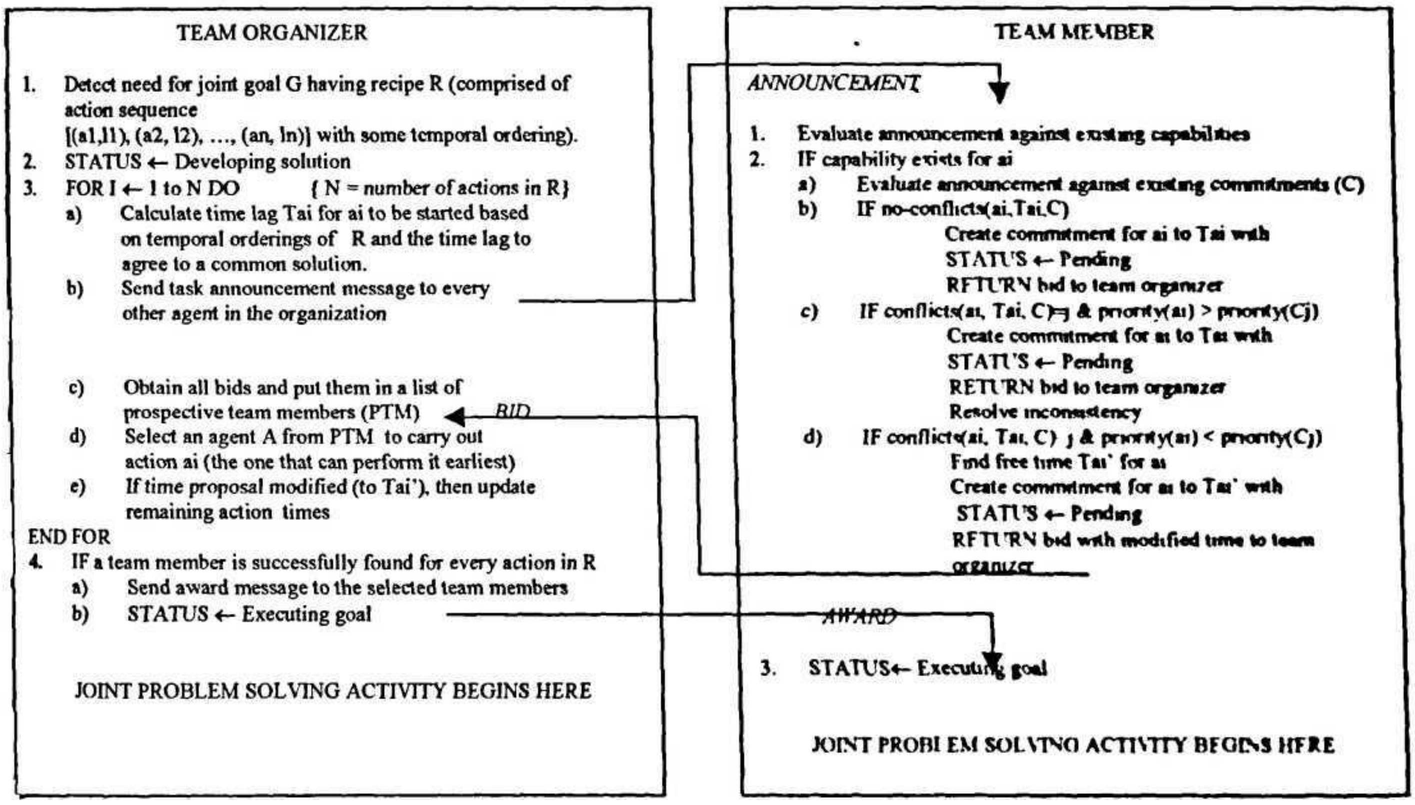
Sender: A_1
Receiver: All team members
Type DECOMMITMENT TO JOINT GOAL
Contents
Joint Goal G_2
Priority
Contribution a_2
Contribution Time
Lower bound on time

Figure 5.11 Decommitment to joint goal message

The task allocation protocol is summarized in Figure 5.12

5.4 Experiment

The inclusion of anytime solutions results in a considerable improvement in the performance of agents This can *be* demonstrated by an experiment (see Figure 5,13) The protocol was implemented in Java and run for two organizations of five agents each. Agents in one organization used anytime algorithms and agents in the other used standard algorithms. Several problem-solving requests were made randomly to each of these organizations, half of which were assumed to have anytime solution. The organizations can handle requests (without decommitments), if they arrive at



```
TEAM ORGANIZER

1.  Detect need for joint goal G having recipe R (comprised of
    action sequence
    [(a1,l1), (a2, l2), ..., (an, ln)] with some temporal ordering).
2.  STATUS ← Developing solution
3.  FOR I ← 1 to N DO          { N = number of actions in R }
    a)  Calculate time lag Tai for ai to be started based
        on temporal orderings of R and the time lag to
        agree to a common solution.
    b)  Send task announcement message to every
        other agent in the organization
    c)  Obtain all bids and put them in a list of
        prospective team members (PTM)
    d)  Select an agent A from PTM to carry out
        action ai (the one that can perform it earliest)
    e)  If time proposal modified (to Tai'), then update
        remaining action times
END FOR
4.  IF a team member is successfully found for every action in R
    a)  Send award message to the selected team members
    b)  STATUS ← Executing goal

JOINT PROBLEM SOLVING ACTIVITY BEGINS HERE
```

```
TEAM MEMBER

ANNOUNCEMENT
1.  Evaluate announcement against existing capabilities
2.  IF capability exists for ai
    a)  Evaluate announcement against existing commitments (C)
    b)  IF no-conflicts(ai,Tai,C)
            Create commitment for ai to Tai with
            STATUS ← Pending
            RETURN bid to team organizer
    c)  IF conflicts(ai, Tai, C)=j & priority(ai) > priority(Cj)
            Create commitment for ai to Tai with
            STATUS ← Pending
            RETURN bid to team organizer
            Resolve inconsistency
    d)  IF conflicts(ai, Tai, C) j & priority(ai) < priority(Cj)
            Find free time Tai' for ai
            Create commitment for ai to Tai' with
            STATUS ← Pending
            RETURN bid with modified time to team
            organizer
AWARD
3.  STATUS ← Executing goal

JOINT PROBLEM SOLVING ACTIVITY BEGINS HERE
```

Figure 5.12 Task Allocation Protocol

Task arrival rate n (every *n* sec)	Anytime algorithm % goals decommited	Standard algorithm %goals decommited
2	46	93
4	40	55
8	18	28
16	0	0
32	0	0

Figure 5.13 Performance of agents using anytime and standard solutions

intervals of 16 or more If the frequency of requests increases, the number of decommitments also increases correspondingly The performance was measured in terms of the goals that were decommited. Since "anytime" algorithms can provide 'some' solutions even in lesser time, the agent can take up other goals if required. As a result, the number of decommitments is far less compared to the organization with standard solutions.

5.5 Conclusions

This chapter described the task allocation protocol Tasks have an associated deadline and priority. They are assigned to agents so that higher priority tasks are executed in time. Since the computational load on any organization of the multi-agent system is unpredictable, a situation could arise where an organization is overloaded, but the multi agent system as a whole has the required resources to take on that load. The resource manager of each organization collects statistics of goals decommited from its agents and performs resource reallocation This is described in the next chapter.

Chapter 6
Dynamic Resource Allocation

The previous chapter described the protocol for allocation of tasks to agents within an organization. When computational load increases, existing agents in an organization may not be able to complete all the goals in time. Hence additional agents may be needed. On the other hand, a drop in the computational load will result in idle agents and lead to inefficient utilization of resources. These two events can occur simultaneously in different organizations of the multi-agent system and cause degradation in its performance.

In order to overcome this situation, the idle agents need to be allocated to the organizations that need them most. What is therefore required is a mechanism for dynamic allocation of agents across the organizations. Section 6.1 highlights the issues not addressed by the existing methods that deal with computational overloads, and shows how they are handled in TRACE. Section 6.2 describes the resource allocation problem. Section 6.3 presents an introduction to market oriented agents. Section 6.4 describes two approaches (price oriented and resource oriented) to market based resource allocation. Section 6.5 gives the details of the proposed method for resource allocation. Section 6.6 reports the results of our simulation experiments. Finally Section 6.7 presents the conclusions.

6.1 Related Work

Execution-time adaptation has been reported earlier in literature [76,77,99,137]. This was described in Chapter 3. For example Ishida et al [137] propose two reorganization primitives, *decomposition* and *composition* for reorganization of multi-agent production systems. Decomposition divides one agent into two and composition combines two agents into one. Decomposition is triggered when the problem solving demand on the system exceeds its ability to respond. Composition is performed when under-utilized resources can be released for use by other systems Triggering of these primitives changes the population of agents and the distribution of knowledge in the multi-agent system.

Our initial work on handling load variations and efficient utilization of resources is presented in [121,123]. This framework, called AASMAn, integrated the contract net protocol with the decomposition and composition primitives described above.

In RETSINA (Reusable Task Structure-based Intelligent Network Agents), developed by Decker and Sycara [72,76,77,99] execution-time adaptation is handled by means of agent cloning. When an agent becomes overloaded, it creates a new agent that is a clone of itself. The clone is set up to use the resources of another processor.

Another solution suggested by researchers is the mobile agent paradigm [146]. A mobile agent is a program that acts on behalf of a user or another program and is able to migrate from host to host on a network under its own control. The agent chooses when and where it will migrate, and may interrupt its own execution and continue elsewhere on the network. The agent returns results and messages in an asynchronous fashion

The aim of these methods is to balance load in order to improve system performance. However, these solutions firstly do not address the details of how resources required to perform decomposition or cloning or agent migration are made available if there are multiple requests for a single resource. A fair allocation is defined as one in which resources are allocated to organizations in direct proportion to their need This need is reflected in the priority of tasks, for the execution of which these resources are required This issue of fairness of resource allocation is crucial, especially in the case of time constrained domains which require allocation of resources to the most critical tasks

The second limitation of these approaches is that they require every agent in the MAS to individually carry out the necessary reasoning for performing decomposition, cloning, or agent migration. The result is inefficient utilization of resources. This is because in situations where a group of agents in the multi-agent system cooperatively solve problems, the need for additional resources needs to be determined for the entire group and not for individual agents of the group.

The proposed resource allocation protocol aims at overcoming these limitations. The first issue is addressed by making use of the economic approach for resource allocation. This simplifies the task of guaranteeing fair allocation of resources. The second limitation is overcome by using one resource manager agent per organization, which determines the resource needs of its entire organization and correspondingly allocates resources.

6.2 Reosurce Allocation Problem

The protocol described in the previous chapter is designed only to perform task allocation given a set of tasks and agents. A mechanism is needed for dynamically allocating agents to organizations based on their problem solving demand. We develop such a mechanism (TRACE-RAP) and combine it with the task allocation protocol in order to obtain a truly adaptive multi-agent system.

Basically, what we have is a market like situation where there is a demand for services (problem solving requests) and agents in the organization supply the required services. Supply and demand are used as the two economic forces to determine the amount of a resource or the supply of services that are provided and its price Under normal conditions the demand can be met by the organizations However, occasionally there may be unpredictable changes in demand, requiring either additional supply of resources or resulting in surplus resources This increase or decrease in the supply of services/ resources, is provided by a proportionate increase or decrease in the number of agents. Our approach is to change the number of agents in the organizations and the knowledge possessed by these agents and thereby *reorganize* the multi-agent system

As the number of agents required is not known a priori, each organization initially starts with a minimum number of permanent agents and a resource manager. Whenever additional agents are required, they are obtained by buying them from markets set up by the resource managers; one resource manager per organization. These resource managers have a set of marketable agents that they wish to sell to the organizations in need of them. The resource managers keep track of the requirements of their respective organizations and dynamically determine how to sell the marketable agents.

The main objective of the resource manager is to sell agents to organizations that require them most (the ones that are executing higher priority tasks). Section 6.4 describes in detail how these markets operate. The agents that are bought from a market are allocated tasks by permanent agents of the organization using the task allocation protocol. The marketable agents serve as contractors or team members and share some of the computational load of the organizations.

6.3 Market Oriented Agents

Various approaches to resource allocation in distributed systems were described in Chapter 3. Of these methods, the one gaining increasing currency is that of a collection of distributed agents as an economic system Projects that have applied market mechanisms to problems in distributed resource allocation include [14,25,68]. In this approach, agents are participants in a distributed computational economy, interacting in the market to further their own interests Behaviors are described in standard economic terms of production, consumption, bidding and exchange.

Wellman suggests that in order to take an economic approach, typically invokes three premises [90]. His first premise is that the fundamental problem to be solved is one of *resource allocation.* Second, that it is useful to model behavior in terms of *rationality abstraction.* And third, that it is essential to consider how authority and activity may be *decentralized.* Each of these is described below.

6.3.1. Resource Allocation

A computer can be viewed as a decision machine, where a decision is about choosing from among potential courses of action to solve a give problem. Every decision - hence every computation is about resource allocation. Choosing to do something entails an allocation of attention and other activity resources to do that thing in lieu of others. Conversely, an allocation of resources defines the activities done and not done. Making such choices appropriately involves weighing the benefits of the activity done against the opportunity cost of foregoing those not done.

Every problem, including ours, can therefore be cast as one of resource allocation. The advantage being that, without considering resources explicitly, it is difficult to express the range of courses of action available, as defined by configurations of resources devoted to the various activities. More importantly, without acknowledging gradations in value, it is impossible to account for tradeoffs among alternate activities [90].

6.3.2 Rationality Abstraction

Economic theory assumes that individual agents are *rational,* acting so as to achieve their most preferred outcome, subject to their knowledge and capabilities. This approach is similar to much work in AI. Newell [2] proposed that a central characteristic of Al practice is a particular abstraction level at which we interpret the behavior of computing machines. Viewing a system at Newell's knowledge level entails attributing to the system, knowledge, goals, and available actions and predicting its behavior based on a *principle of rationality* that specifies how these elements dictate action selection. Newell's rationality principle is:

> *If an agent has knowledge that one of its actions will lead to one of its goals, then the agent will select that action.*

This formulation relegates all matters of resource allocation and graded preferences to some ad hoc auxiliary principles From the economic perspective, a satisfactory comprehensive rationality principle should address choice among alternate activities and resource allocations that accomplish goals to varying degrees. Thus, some coherence based rationality principle is required to make sense of the sorts of agent attitudes - knowledge, belief, preference, intention - commonly used in multi-agent system research.

6.3.3 Decentralization

Within economics, the problem of synthesizing an interaction protocol via which rational agents achieve a socially desirable end is called *mechanism design.* This is exactly the problem we face in designing multi-agent systems.

As all the three premises hold good for our problem we exploit existing economic ideas.

The advantage of using economic principles for resource allocation is that many ideas and results from economics can be directly applied instead of developing new theories. These methods have also been found to possess the properties of feasibility, monotonicity (i.e., its quality of solutions improves with time) and fast convergence and can therefore be used in the development of real time distributed systems [8,51,68],

6.4 Market Based Approaches to Resource Allocation

In our human society, resource allocations are in most cases performed through markets. This occurs on many different levels and in many different scales, from our daily grocery shopping to large trades between big companies and / or nations The market approach to resource allocation in human society has inspired the multi-agent system community to construct similar concepts for multi-agent systems, where trade is performed between computational agents on *computational markets* Wellman refers to this as market oriented programming [89].

In computational markets, a common approach is to use a mechanism that obtains *general equilibrium* General equilibrium is obtained when a set of prices (one price for each commodity) is found such that supply meets demand for each commodity and where the agents optimize their use of resource at the current price level. In virtually all multi-agent systems there exist some scarce resources. Thus the issue of resource allocation is of fundamental importance. Two basic microeconomic approaches towards developing distributed resource allocation mechanisms based on general equilibrium theory *are price directed approach* and *resource directed approach* [8,38].

6.4.1 Price Directed Approach

In the price directed approach [8], an initial allocation of resources is made and an arbitrary set of system wide initial prices is chosen. Prices are then iteratively changed to accommodate the *demands* for resources until the total demand for a resource exactly equals the amount available. At this point, the resulting final allocation of resources is pareto optimal. Pareto optimum condition is one in which no one agent can be made better off without making someone else worse off.

The market equilibrium is given by [8]

$$z(p) = 0 \qquad (1)$$

where $z(p) = [z(p_,), z(p_2), \dots z(p_k], z(p_1)$ being the aggregate excess demand for commodity i $p = [p_1, p_2, \dots p_k]$ where $p,$ is the price for commodity i, and k is the number of commodities. The aggregate excess demand for commodity i, at price p_i, is the sum of the supply and demand of all agents, i.e.

$$z(p_i) = \sum z_a(p_i)$$

where $z_a(p_i)$ is the demand of agent a for resource i at price p_i. The demand of an agent describes how much an agent is willing to buy (or sell - a negative demand) at a specific price level. In the price-oriented scheme the price vector is updated iteratively, until equation *1* is fulfilled. Since prices are only relative, p_k is set equal to *1* and only *k*-

1 elements are searched in the price vector. Inputs to this scheme are the respective net demands of each agent, $z_a(p)$, where a denotes an agent. WALRAS, developed by Wellman [89], is a prototype environment for specifying and simulating computational markets.

6.4.2 Resource Directed Approach

An alternative way to express the general equilibrium is to define it as an allocation such that, for each commodity, each agent's marginal utility is the same for an additional amount of resource. In this approach [37,38,68], during each iteration, each agent computes the marginal value of each resource it requires given its current allocation of resources (i.e. computes the partial derivative of its utility function - performance) with respect to that resource, evaluated at the current allocation level. These marginal values are then sent to other agents requiring use of this resource. The allocation of the resource is then changed such that agents with an above average marginal utility receive more of this resource and agents with a below average marginal utility are allocated less of the resource. When analytic formulas are used to compute the performance realized by a given resource allocation, an actual reallocation need not (but may) take place immediately after each iteration; an agent may simply compute its new allocation at each iteration and the resources may then be allocated whenever the algorithm is terminated. In the case that actual measurements are used, however, resources must be immediately reallocated in order for each agent to measure its performance under the new allocation. Applications like the distributed file allocation problem [68], and power load management [38] use the resource directed approach to develop computational markets.

6.4.3 Discussion

The differences between these two approaches are that firstly, their inputs are different. In the price directed scheme, demand is the input and in the resource directed case the inputs are some derivatives of the utility function. In standard micro-economic theory, the utility function is the primary concept and the demand is derived from the utility function. Another important difference between the price directed algorithm and resource directed algorithm is the *fairness versus feasibility* of the solution they produce [8]. For the price oriented algorithm to converge in reasonable time, the termination condition is $(|z(p)| < \varepsilon)$ instead of $(z(p) = 0)$. For the resource oriented algorithm to converge, the termination condition is (marginal utility $< \varepsilon$) for all agents. In the resource oriented case this means that the allocation is not perfectly *fair*, i.e., some agents pay less than they would have done on a perfect market (marginal utility = 0 for all agents), while others will pay more. On the other hand, in the price-oriented case, the allocation is not perfectly *feasible* (the total amount of resources allocated equals the amount available). In this thesis, we focus on price oriented algorithm in order to obtain a fair allocation.

6.5 Resource Allocation Protocol

The multi-agent system organization for TRACE was described in Chapter 4. The system consists of a collection of organizations that in turn consist of a set of agents that cooperate with each other to achieve goals. The agents within an organization exhibit team rationality by obeying the joint responsibility code of conduct for joint actions [96].

Initially it is only the permanent agents that comprise an organization. As problem-solving activity progresses, organizations go through variations in load. In the event of a computational overload, the TAP accommodates high priority tasks by decommiting low priority ones. In order to minimize these lost requests, marketable agents need to be allocated dynamically to the organizations in accordance with their computational loads. This allocation of resources, which results in reorganization of the multi-agent system, is done by the resource allocation protocol (RAP). We assume that requests that are once decommited will be requested again. The RAP reorganizes the multi-agent system so that these decommited requests can be honored when they arrive again.

The resource manager obtains the resource needs of an organization from its permanent agents, and on the basis of this information, arrives at a suitable allocation of marketable agents. As the number of marketable agents is fixed, and multiple organizations could be contending for these agents, an allocation is arrived at on the basis of the criticality of decommited tasks, for the execution of which these agents are required. The permanent agents of an organization convey information about the criticality of decommited tasks indirectly by contributing some funds to the resource manager; the more the criticality of decommitments, the higher the contribution of funds. The permanent agents also specify how many additional agents (μ) would be required by the organization. The method used for obtaining μ is explained in Section 6.5.1. Thus the contribution of funds made by an organization indicates the maximum price that the organization is willing to pay in order to buy μ marketable agents. The contribution of funds varies from organization to organization and reflects their relative needs for additional resources. The organizations that offer more funds per agent are considered to be more in need of resources than the ones offering less.

The resource managers periodically determine the resource needs of their respective organizations and accordingly conduct reorganization. Each such period is called a *reorganization cycle.* Thus it is not necessary for every agent of an organization to participate in the process of determining a suitable allocation.

We therefore have the multi agent system organized as a *market economy* composed of interacting buyers and sellers. The commodities in this economy are processing resources (marketable agents) required to achieve goals. *Buyers* are organizations that wish to purchase new agents in order to perform some computation. *Sellers* are the resource managers that wish to sell the marketable agents for the duration of one reorganization cycle. In this economy, monetary funds encapsulate resource rights, and *price* equates the supply and demand of processing resources. The buyers and sellers execute a resource allocation protocol to arrive at an optimal allocation of resources. Reallocation of resources is done at the beginning of every reorganization cycle and results in a reorganization (change in the number of agents in the organizations, their communication structure and the distribution of knowledge) of the multi-agent system. For reallocation to be completed, each resource manager goes through the following steps:

1. Collects statistics from agents of its organization.
2. Computes the equilibrium allocation.
3. Notifies permanent agents of its organization about the new allocation.

There are two possibilities with regard to the kind of resources to be sold. The resources could all be of the same kind (homogenous) or there could be resources of different kinds (heterogeneous). In terms of the agent architecture described in Chapter 4, agents are said to be homogenous if all of them have the same kind of goal processor. In this case every agent has the capability to execute every task required by its organization (if it has the recipe). On the other hand, agents are considered heterogeneous if they possess different kinds of goal processors. The capabilities of all the agents are therefore not the same in this case.

The allocation of homogenous resources is described first and then this is extended to handle heterogeneous resources.

6.5.1 Allocation of Homogenous Resources

In order for proper allocation to take place, the requirement for resources in any reorganization cycle is determined on the basis of the information about the previous one All agents convey the following *four items of information* (about the previous reorganization cycle) to their respective resource managers at the beginning of every reorganization cycle:

1. Information about the number of decommitments (D)

This information is sent by the permanent agents because only they act as organizers and keep track of the number of decommitments. It is assumed that the demand for new agents, μ, in an organization can be easily computed from the number of decomitments (D) made by the organization. If an agent is capable of completing, on an average, G goals per reorganization cycle, the number of new agents that are required is D/G. As the requests that are decommited by the task allocation protocol are the ones that have low priority, they are very much likely to occur again. Consider the example where agents are involved in information handling for users. User requests that are decommited are the low priority ones and there is a high probability of the user resubmitting his previously rejected request. Thus the number of decommitments can be used as a reasonable measure of resource requirements, and the type of requests decommited provide information about the required capabilities. Based on this information about decommitments, new agents are introduced into an organization that have the capability to take on the decommited goals when they are requested again.

2. Information about the decommited goals.

This information is also sent by the permanent agents and is used for dynamic distribution of domain knowledge to agents. The type of requests decommited provide information about the required capabilities. New agents are introduced into an organization after transferring the domain knowledge (i.e. the recipes required for executing the decommited goals) to them. The transfer is made by the resource manager which has the complete domain knowledge for executing all the goals required by its organization. These new agents can therefore take on the previously decommited goals when they are requested again.

3. Information about the idle time.

This information is conveyed by all marketable agents (because only these agents can be reallocated, not the permanent ones) and helps in identifying idle agents.

Marketable agents that remain idle for more than fifty percent of the reorganization cycle time can be treated as superfluous and considered for allocation to some other organization in need of them. Thus if an organization is allocated X marketable agents in a cycle, has D (from item 1) equal to zero for that cycle, and has Y agents that remain idle most of the time, then the number of agents it requires for the next cycle, μ, is taken to be X-Y.

4. Information about the contribution of funds (F).

Permanent agents in an organization contribute funds to their resource manager in every reorganization cycle. The sum of these values for all permanent agents indicates the maximum the organization is willing to pay for buying μ (obtained from item 1 or 3) additional agents.

The funds contributed by different organizations reflect their relative needs. The organizations that contribute more are deemed to be more in need of additional agents than the ones contributing less. The allocation of resources is done on the basis of the amount of funds contributed. Thus the organization that makes the highest contribution is allocated resources first.

We assume that the amount of funds to be contributed is determined by the application. The monetary funding units are used as an abstract form of priority in the multi-agent system. It is the applications burden to ensure that important computations are well funded.

Each resource manager conducts markets on behalf of high level applications and intimates the equilibrium price of a marketable agent to the permanent agents of its organization. On the basis of its funding, the equilibrium price and the number of decommitments in any cycle, the application can determine how much to contribute for the next cycle. We feel that a high level application should not be encumbered with decision making at the low level market mechanisms that locate and purchase the resources necessary for its execution. At the same time, however, it should be possible for an application to exert some control over the general allocation of funds. The proposed method provides a uniform mechanism with these capabilities.

Every resource manager encapsulates details about the funds and demand for new agents for its organization and communicates this information to every other resource manager of the multi-agent system. The protocol consists of this communication step followed by a local computation by each resource manager. Each resource manager locally computes the equilibrium price of an agent. The demand of organization *a* at price *p*, $z_a(p)$, indicates how much it is willing to buy at price *p*. The total demand for agents across all the organizations is $\sum_{a=1}^{n} z_a(p)$, where *n* is the number of organizations in the multi-agent system. Let $s(p)$ be the supply of marketable agents at price *p*. The market will be in equilibrium when p has a value such that

$$\sum_{a=1}^{n} z_a(p) = s(p) \qquad (2)$$

In order to find this price, the resource managers initialize *p* to the maximum of prices offered by all the organizations. This price is then iteratively changed till equation 2 is satisfied. The number of iterations can however be reduced by using an approximation condition of the form

$$\left|\sum_{a=1}^{n} z_a(p) - s(p)\right| \leq \varepsilon$$

where $\sum_{a=1}^{n} Z_a(p) - s(p)$ denotes the aggregate excess demand

6.5.2 Equilibrium Price Computation

A more precise statement of the computation is now given. We use the following notation

F_i - denotes the contribution of funds made by the organization /.
minprice - indicates the minimum price at which the resource managers can sell the marketable agents

The demand for marketable agents at organization a at price *p* is given by the function $z_a(p)$, which is defined as

$$z_a(p) = \begin{cases} \mu_a & \text{if} \quad p < F_a/\mu_a \\ F_a/p & \text{otherwise.} \end{cases}$$

The total supply of marketable agents at price *p* is given by the function *s(p),* which is defined as

$$s(p) = \begin{cases} \text{total number of marketable agents in the MAS} & \text{ifp } \geq minprice \\ 0 & \text{otherwise.} \end{cases}$$

The market is in equilibrium when $|\sum_{a=1}^{n} z_a(p)-s(p)\backslash \quad \leq \varepsilon$.

Each resource manager goes through the following computations:

1. Communicates F and m for its organization to every other resource manager
2. Intialization
   The equilibrium price p is initialized to the maximum of prices offered by all the organizations and the excess demand *z(p)* at price p is evaluated.
3. Iteration
   a) while $(\backslash z(p)\backslash > \varepsilon)$ and $(p > minprice)$ do
   b) decrement price by a small amount *p'.* this is referred to as the step size parameter, $p = p-p'$

Once the equilibrium price is determined, an optimal allocation of resources is found The resource managers then notify all permanent agents of their organization about the new allocation. The permanent agents accordingly update their organizational knowledge.

6.5.3 Allocation or Heterogeneous Resources

The above algorithm can be easily extended to perform allocation of resources that are heterogeneous. Let there be *k* types of resources, and *p - [p1, p2, ..., pk* be the price vector, where p_i denotes the price of resource *i*. The a^{th} organizations demand for resource i at price pi, $z_a(p_i)$, describes how much of resource *i* the organization will buy at price p_i. The total demand for resource i across all the organizations is $\sum_{a=1}^{n} z_a(p_i)$, where *n* is the number of organizations in the multi-agent system. The price of resource i should be fixed such that the total demand equals the supply, i.e.,

$$\sum_{a=1}^{n} Za(pi) = Si(pi) \qquad (3)$$

The market will be in equilibrium when a price vector is found for which the above equation is satisfied for all types of resources, i.e.,

$$\sum_{a=1}^{n} z_a(p_i) = S_i(P_i) \quad for\ i = I..\ k$$

Thus the prices in the price vector need to be iteratively changed till equation 3 is satisfied for all types of resources. However an approximation of the form

$$|\sum_{a=1}^{n} z_a(p_i) - s_i(p_i)| \leq \varepsilon$$

can be used to terminate the iterations in reasonable time.

In order to achieve equilibrium, every agent of an organization conveys the same information (items 1, 2, 3 and 4) as in the case of homogeneous resources to the resource manager.

From items I and 3 the resource manager determines the number of agents of each type required by its organization. The amount of funds indicated in item 4 is the total contribution made for buying all types of agents. This total contribution now needs to be split among the different types of agents. The funds are split in the combined ratio of the minimum prices for these agents and the number of agents required of each type.

If T is the total contribution of funds and two types of agents A_1 and A_2 are required, then,

allocation of funds for buying A_1, $F_{A1} = T*p_a*\mu_{A1}/(p_a *\mu_{A1}+p_b*\mu_{A2})$ and

allocation of funds for buying A_2, F_{A2} - $T*p_b*\mu_{A2}/(p_a*\mu_{A1}{}^{+}p_b*\mu_{A2})$,

where p_a and p_b are the minimum prices at which the resource managers can sell A_1 A_2 and μ_{A1} and $m_{A2}$are the number of agents of type A_1 and A_2 respectively.

In this way the resource managers obtain the information about the requirement for different types of agents, the number of agents of each type, and the funds associated with each type of agent. The above algorithm (for homogenous resources) can now be applied to each type of resource.

6.5.4 Dynamic Distribution of Knowledge

The protocol described above allocates marketable agents to organizations. These agents can however lack the domain knowledge required to take on goals of an organization. This means that such agents need to be first endowed with the required knowledge before they are allocated to an organization. In addition to managing resource allocation, the resource manager also does the job of allocating this knowledge (recipes) to the new agents. We assume the knowledge to be available in a form that facilitates this kind of distribution.

The resource manager possesses all the knowledge required by its organization. Out of this entire knowledge only a selected portion is allocated to the new agent. In order to determine this portion the resource manager obtains information about the decommited goals from all agents of its organization. The domain knowledge required to execute these goals is then transferred to the incoming agent.

This kind of dynamic distribution of knowledge enables effective use of available computational resources; an agent that is idle but lacks knowledge required to execute goals can acquire that information as indicated above. This also means that agents do not have to be preloaded with extensive amounts of knowledge which may or may not prove useful.

6.6 Experiments

In order to evaluate the effectiveness of the proposed mechanism a number of experiments were carried out. These serve to quantify its ability to

- *Reduce the number of decommitments.*
- *Fairly distribute resources among competing goals.*
- *Adapt to changes in computational load by reorganizing the multi-agent system.*
- *Make effective use of resources.*

In addition to this the MAS possesses the property of openness New agents can be added to an organization by advertising it with the resource manager of that organization. Entire organizations can also be added by advertising the availability of the resource manager of the new organization with all the existing resource managers

The system was simulated in C language and the behavior of the system was studied by randomly varying the computational load at different organizations of the multi-agent system. These studies were done assuming that the resources are homogenous.

6.6.1. Reduction in Decommitments

The first experiment was done to measure the reduction in the number of decommitments made by the system. Each organization of the multi-agent system was assumed to have 10 permanent agents and the number of marketable agents was 10 times the number of organizations.

The system was allowed to run for 100 reorganization cycles by randomly varying the computational load in every reorganization cycle. Different organizations contributed different amounts of funds but the amount contributed by an organization was held constant over all the 100 reorganization cycles. The total number of decommitments over the entire run was found. The experiment was then repeated for a multi-agent

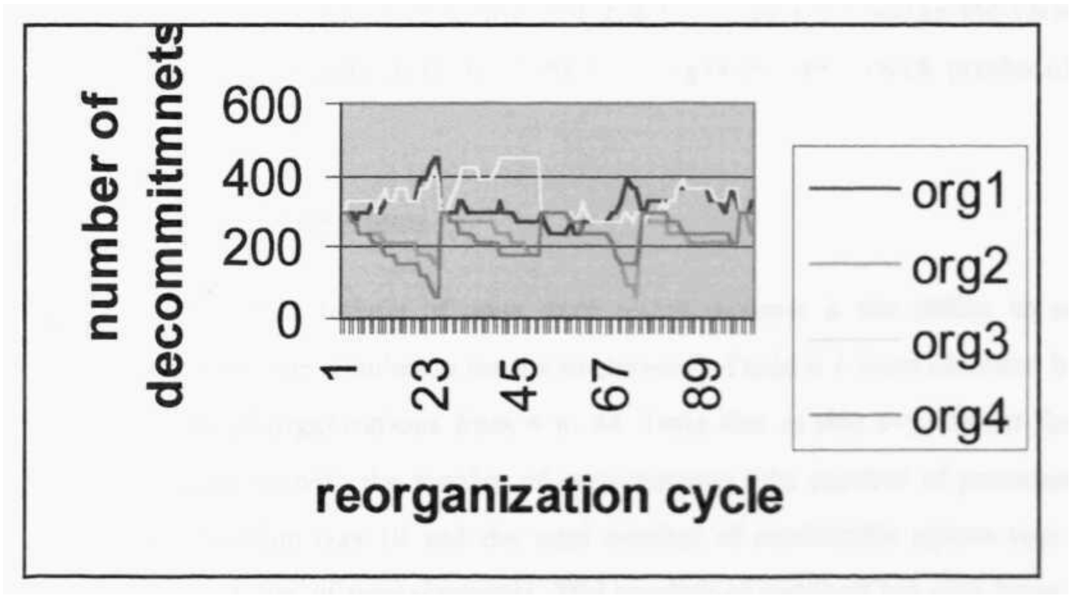


Fig 6.1. Variation in decommitments over 100 cycles in 4 organizations

Number of Organizations	Percentage reduction in decommitments in the MAS
4	74
8	80.8
16	80.2
32	76.2
64	76.2

Table 6.1 Percentage reduction in decommitments

system without using TRACE-RAP (i.e. by equally dividing the marketable agents among the organizations and keeping the number of agents in each organization always constant). From these two results the percentage reduction in the number of decommitments using the reorganization method was determined. The results of this study are summarized in Table 6.1. The graph in Figure 6.1 shows the variations in the requirement for agents in each of the four organizations, which produced the results given in Table 6.1.

Scaling to Larger Systems

A desirable characteristic of open multi-agent systems is the ability to scale well to large systems. The simulation results as shown in Table 6 1 were obtained by increasing the number of organizations from 4 to 64 Note that in this experiment the number of agents increases with the number of organizations (the number of permanent agents in each organization was 10 and the total number of marketable agents was taken as 10 times the number of organizations) The number of requests has also been increased in the same proportion. Basically, the conditions to which an organization is subjected are the same. But the number of such organizations has been scaled up. This increase however did not effect the percentage reduction in decommitments This indicates that with respect to reduction in decommitments, the proposed reorganization approach scales well to large systems.

These results were obtained in the absence of a funding strategy. The performance of the system can however be improved by having the application make use of effective funding strategies.

6.6.2 Fairness of Resource Allocation

	Orgl	Org2	Org3	Org4
Funding Ratio	33.23	23.33	20.00	23.22
Ratio of agents allocated	34.21	23.68	18.42	23.68

Table 6.2 Fairness of resource allocation for a MAS with 4 organizations

	Orgl	Org2	Org3	Org4	Org5	Org6	Org7	Org8
Funding ratio	19.57	1.09	9.78	19.57	19.57	1.09	9.78	19.57
Ratio of agents allocated	18.75	1.25	11.25	18.75	18.75	1.25	11.25	18.75

Table 6.3 Fairness of resource allocation for a MAS with 8 organizations

Organization	Funding ratio	Ratio of agents allocated
1	9.78	9.38
2	0.54	Q.62
3	4.89	5.62
4	9.78	9.38
5	9.78	9.38
6	0.54	062
7	4.89	5.62
8	9.78	9.38
9	9.78	938
10	0.54	062
11	4.89	562
12	9.78	938
13	9.78	938
14	0.54	062
15	489	562
16	9.78	938

Table 6.4 Fairness of resource allocation for a MAS with 16 organizations

In TRACE, funds abstractly encapsulate relative resource rights, and are analogous to priority. The funding units are abstract since they are completely independent of resource details. They are also relative since the amount of resource to which an organization with a given amount of funding is entitled, varies dynamically in proportion to the contention for that resource.

In order to test the fairness of resource distribution a set of experiments was done for different funding ratios among organizations. The results of these experiments are summarized in Table 6.2. The first row specifies the funding ratio of organizations. The second row indicates the relative number of agents obtained by each organization, A fair distribution is one in which each organization is able to obtain a share of resources that is close to its share of total system funding. As we can see from the Table 6.2, TRACE allocates resources in a manner that is reasonably close to the funding ratio in all the runs. Tables 6.3 and 6.4 present representative simulation runs which demonstrate that a reasonable degree of fairness is continuously maintained even in large systems.

6.6.3 Adaptiveness of the Multi-agent System

The main objective of TRACE is to make the multi-agent system adaptive to variations in computational load. Figure 6.2 shows the variation in number of decommitments over 100 reorganization cycles and Figure 6.3 shows new agents acquired by an organization over 100 reorganization cycles. Initially the number of agents is 10 in all organizations (the number of permanent agents). As the number of decommitments in a reorganization cycle increases, the number of new agents in the next reorganization cycle increases correspondingly. For instance in reorganization cycle 25 the number of decommitments increased to 300. As a consequence of this, the number of marketable agents in the next cycle increased to 10 (for G=30). Similarly it can be seen that a decrease in the number of decommitments results in a decrease in the number of agents. These results demonstrate the ability of the multi-agent system to adapt to computational load variations, thereby making it applicable to time constrained domains.

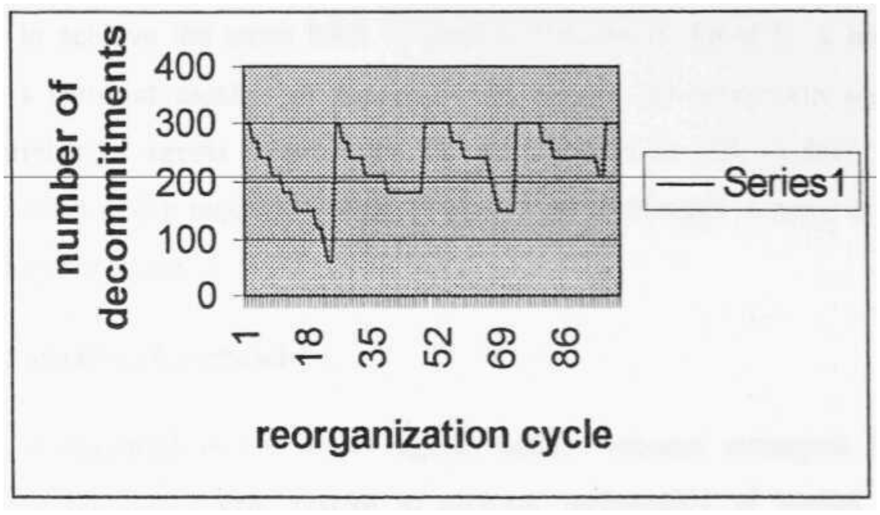


Figure 6.2 Variation in the number of decommitments

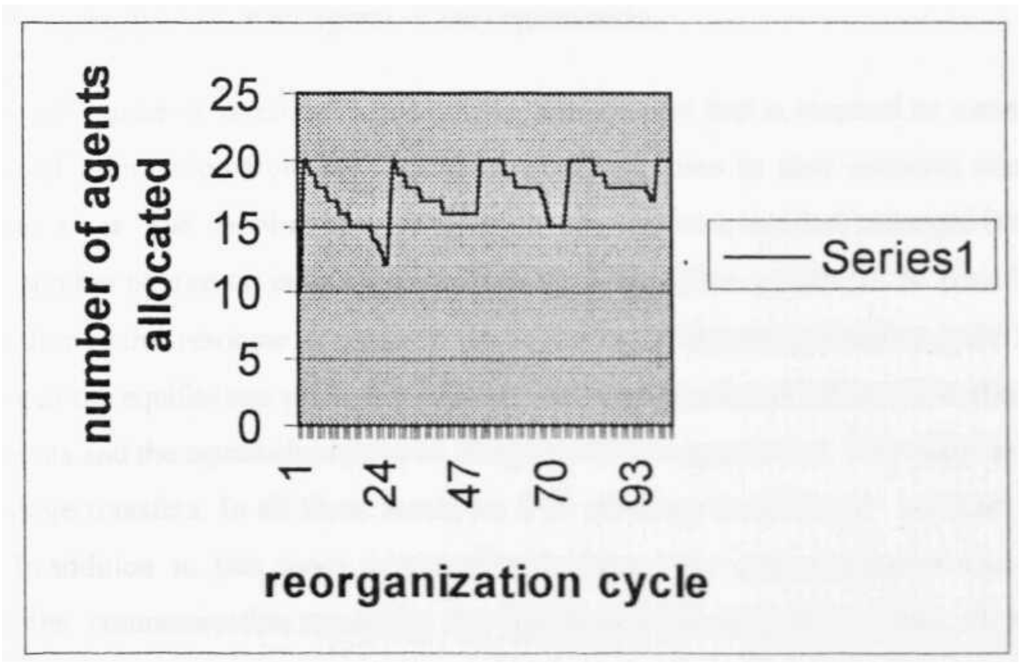


Fig 6.3. Variation in the number of marketable agents allocated

6.6.4 Efficient use of Resources

As shown in Figure 6.3, the average number of agents over 100 reorganization cycles is 17. In order to achieve the same level of performance as in TRACE, a multi-agent system with a constant number of agents would require 20 permanent agents (the maximum number of agents required by the organization in 100 cycles). Thus the proposed approach which requires around 17 agents on an average is more economical in terms of resource usage.

6.6.5 Reorganization Overhead

One source of overhead is the set of agents called resource managers that were introduced into the multi-agent system to perform reallocation of agents. The sole function of these agents is to perform reallocation at the beginning of every reorganization cycle. The resource managers however remain idle for the remaining part of the cycle. In order to make effective use of these resource managers, they can be allocated tasks just like other agents of the organization.

The second source of overhead is the communication cost that is incurred as a result of transfer of information from the agents of an organization to their resource manager and vice versa. The number of messages that are sent to a resource manager is equal to the number of agents in its organization (N). Thus there would be N transfers of information to the resource manager at the beginning of the reorganization cycle. After arriving at the equilibrium price, the resource manager broadcasts information about the new agents and the equilibrium price to all agents of its organization. This takes another N message transfers. In all there would be 2*N message transfers per reorganization cycle. In addition to this some communication takes place among resource managers. This is the communication regarding the funds and the required number of agents that is broadcast by every resource manager to every other resource manager. However this communication is not considerable since the number of resource managers is very small compared to the total number of agents in the multi-agent system.

The third factor that needs to be considered is the time required by resource managers to arrive at the equilibrium price i.e., when

$$|Z(p)| < \varepsilon.$$

The convergence time in general depends on two values: the number of iterations required to reach equilibrium, and *n* (the number of buyers and sellers). Let us consider the number of iterations first. The number of iterations required for convergence depends on *p*' (the step size parameter), and e. The smaller the values of *p*' and e, the more feasible is the allocation of agents, but larger is the number of iterations. Larger values of *p*' and e reduce the number of iterations but may not result in an allocation with the same degree of feasibility. The price of an agent that resource managers arrive at may not be sufficiently close to the actual equilibrium price. This results in an infeasible allocation of resources where the amount of resources allocated may not be equal to the amount of resources available.

The second value that determines the convergence time is *n*. In general, in multi-agent systems that use economic approach n equals the total number of buyers and sellers. However in TRACE, *n* is equal to the number of resource mangers since they represent the resource needs of the entire organization. Thus *n* is significantly reduced and therefore results in faster convergence.

6.7 Conclusions

Techniques for building multi-agent systems that can adapt to changing environmental conditions are of great interest. This chapter described a protocol, TRACE-RAP, for resource reallocation and presented the simulation results. The main objective of the proposed mechanism is to obtain a multi-agent system that adapts to varying computational loads and can therefore be used for time constrained applications. Our objective is achieved by using the economic approach for reallocation of resources. This method is used in conjunction with the task allocation protocol, TRACE-TAP, described in chapter 5.

There are many advantages of using the microeconomic approach for controlling resource usage in a distributed system. Firstly, it allows direct application of many ideas and results instead of developing new theories. Secondly, it is simple to implement. Thirdly, these methods possess the properties of monotonicity, feasibility, and fast convergence. Such an approach therefore holds great promise in that it provides a single decentralized framework, which reduces the complexity of designing large, distributed systems.

Chapter 7
Conclusions and Future Work

This is the concluding chapter of this thesis and summarizes the main contributions and the methodology used in this research. Finally some areas for further work are presented.

7.1 Summary and Methodological Issues

The rapid proliferation of multi-agent system applications, like management of networks for electricity transport and telecommunication, in factories to control manufacturing processes, in the medical domain for monitoring the condition of patients in an ICU, etc, places increased demands on the multi-agent system builder [97]. The multi-agent systems of today are expected to operate in increasingly complicated environments that are dynamic and unpredictable. The type and frequency of requests in these applications varies non-deterministically. Consequently it is becoming increasingly important to address the issue of adaptability of multi-agent systems to changing environmental conditions.

This research started with the aim of developing an adaptive organizational policy for multi-agent systems that are targeted towards soft real-time domains. Our intention was to obtain a robust multi-agent system that could withstand computational load variations. As agents operate in dynamic and unpredictable environments, where their beliefs and goals are neither correct nor complete, it is certainly necessary for each agent to be endowed with team rationality [96] This helps in keeping the amount of wasted resources to a minimum. In order to cater to time constrained domains, there also has to be a means of preempting lower priority tasks in preference to higher priority ones.

However, it is not enough if individual agents possess team rationality and the ability to preempt tasks; there should also be a means of handling computational load variations. This is possible only if the entire multi-agent system has the ability to dynamically change its organization to suit the existing problem solving requirements. This reorganization should be done so that the demand for agents with a certain skill always matches the supply of agents having that skill. In case of overloads, preference has to be given to higher priority tasks.

Existing methods for implementing organizational policies [8,24,62,68,71,111], address only some of the above mentioned issues in isolation and hence cannot meet all the requirements of dynamic adaptation. This thesis proposes a single comprehensive organizational policy (TRACE) that can operate under time constraints and varying computational loads.

The entire MAS is viewed as consisting of several problem-solving organizations. Each organization in turn consists of multiple agents and a resource manager. Problem solving requests with an associated deadline arrive at the agents of these organizations. A request that arrives at an organization is solved cooperatively by agents within that organization and independently of the other organizations. The rate of arrival of problem solving requests at each of these organizations varies with time. As a result of this variation, the requirement for resources at each organization also keeps changing. At any particular instant, some organizations may have additional resources, while others may need more resources resulting in some requests that cannot be completed in time. In order to minimize these lost requests, the resource managers dynamically reallocate resources to organizations so as to balance the demand for resources with its supply. This is done by means of a market-oriented protocol. Whenever reallocation is done, the most critical tasks are allocated resources first.

Following a layered approach, the problem of designing such an organizational policy, is divided into the following two sub-problems.

1. Allocation of tasks to agents within an organization through the task allocation protocol (TAP), and

2. Allocation of resources to each of these organizations, through the resource allocation protocol (RAP).

The agents in each organization cooperatively process problem-solving requests by making use of the TAP. At regular intervals of time (called reorganization cycle), these agents report statistics of the general pattern of requests, the number of decommitments, the percentage idle time, and an indication of the criticality of the decommited requests through funds. The resource managers apply the RAP to this information and reallocate resources. This causes an increase/decrease in the number of agents in the organizations and the distribution of knowledge to agents, and thereby reorganizes the multi-agent system.

To evaluate the effectiveness of the proposed approach simulated experiments were done. The behavior of the system was studied under varying problem solving demands. This was compared with the behavior of a MAS with fixed number of agents. On the basis of these experiments it is demonstrated that the proposed protocol possesses the following properties:

- Allows agents to

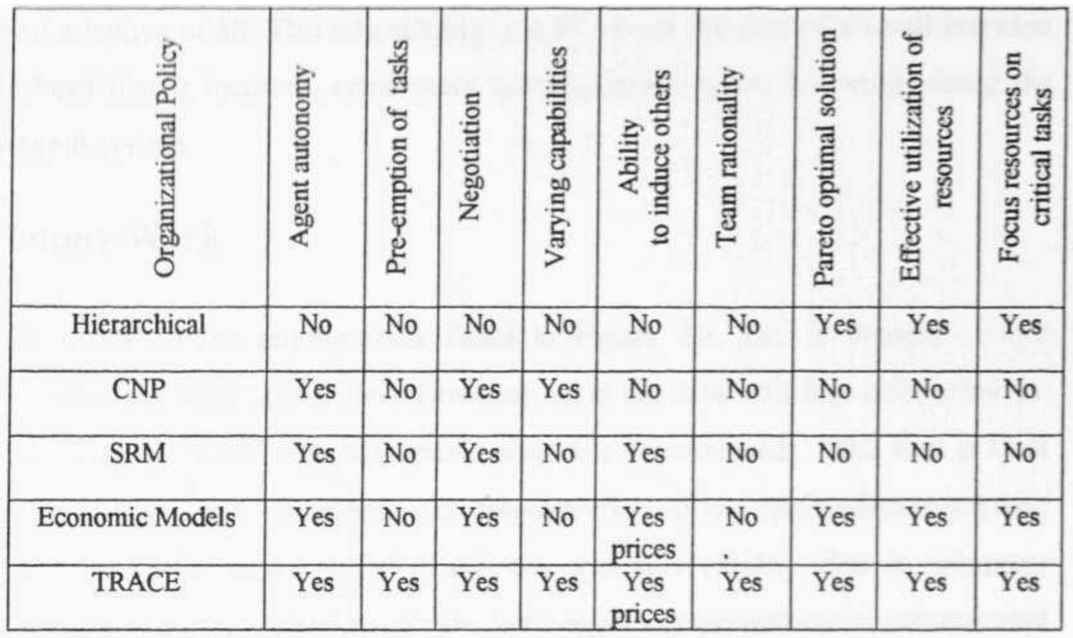
    i) Adapt to unpredictable changes in problem solving environment (by keeping its beliefs and goals always consistent with the latest information that it receives from the environment/ other agents).

    ii) Focus on higher priority tasks.

- Allows the multi-agent system to

    i) Adapt to changes in load by diverting resources where they are needed most

    ii) Add new agents for problem solving in an incremental manner.



Organizational Policy	Agent autonomy	Pre-emption of tasks	Negotiation	Varying capabilities	Ability to induce others	Team rationality	Pareto optimal solution	Effective utilization of resources	Focus resources on critical tasks
Hierarchical	No	No	No	No	No	No	Yes	Yes	Yes
CNP	Yes	No	Yes	Yes	No	No	No	No	No
SRM	Yes	No	Yes	No	Yes	No	No	No	No
Economic Models	Yes	No	Yes	No	Yes prices	No	Yes	Yes	Yes
TRACE	Yes	Yes	Yes	Yes	Yes prices	Yes	Yes	Yes	Yes

Figure 7.1 Comparative Study

Organizational Policy	Maintaining models of other agents	Communication
Hierarchical	Yes	Direct addressing
CNP	Yes No	Directed addressing Broadcast
SRM	Yes	Directed addressing
Economic Models	No	Broadcast
TRACE	No	Broadcast within an org for task allocation and additional communication for resource allocation once every reorganization cycle

Figure 7.2 Comparison of Overheads

Figure 7.1 shows a comparison of TRACE with the existing mechanisms Figure 7.2 compares the overhead associated with all these methods As can be seen from these figures, TRACE possesses all the required features and therefore can be said to be the most adaptive of all. This adaptability is achieved at the cost of a small increase in overhead that is incurred, once every reorganization cycle, for reorganizing the multi-agent system.

7.2 Future Work

TRACE meets all the requirements listed in Figure 7.1, and is therefore more adaptive than the other approaches. However, as is the case with any computational solution, this mechanism also has certain areas for improvement. The first is fault tolerance. In TRACE it is currently the responsibility of the application to recover from failures. In future we intend to achieve goal survivability, that is, whatever might happen to the individual agent, the multi-agent system makes the commitment to meet the goal. Consequently, TRACE should ensure task reassignment to accomplish the task.

Another issue not addressed in this thesis is multi-level contracting. We allowed single level contracting, where a single team organizer and some team members engage themselves in joint problem solving. This framework could be extended, by having team members delegate part of the job, allocated to them by their organizer, to other agents and in turn become organizers for those parts. We could then apply this framework to supply chains by having both linear and tree like organizer - contractor relationships.

In addition to this we propose to work towards dynamically varying the reorganization cycle time. Since the environment is unpredictable, having a fixed cycle time may not be acceptable always The introduction of some means of dynamically varying the reorganization cycle time will make the framework more adaptive.

In TRACE, allocation of resources is done using the price oriented approach This results in allocations that are fair but not perfectly feasible. In future we will incorporate the resource directed approach also into this framework. This will provide the application with an option of selecting one of these two methods depending on its desire to achieve fairness or feasibility.

References and Bibliography

[1] AH Bond and L Gasser (eds), "Readings in Distributed Artificial Intelligence", Morgan Kaufmann, 1988.

[2] A Newell, "The Knowledge level", Artificial Intelligence , 18, pp 87-127, 1982.

[3] A S Rao, M P Georgeff, "BDI Agents: From Theory to Practice", Proc. of First International Conference on Multi Agent Systems, ICMAS-95, San Fransisco CA, pp 312-319, 1995.

[4] A S Rao, M P Georgeff, E A Sonenberg, "Social Plans: A Preliminary Report", Proc. Modelling An Autonomous Agent in Multi-Agent World, 1991.

[5] Ahmed Karmouch, "Mobile Software Agents for Telecommunications", Guest Editorial, IEEE Communications Magazine, pp 24-25, July 1998.

[6] Alan Garvey, Victor Lesser, "A Survey of Research in Deliberative Real-time Artificial Intelligence", Real Time Systems, 6, 3, pp 317-347, 1994.

[7] Anne Collinot, Barbara Hayes Roth, "Real-time Performance of Intelligent Autonomous Agents", In Decentralized AI-3, Editors Eric Werner & Y Demazeau, Elsevier Science Publishers, pp 341-356, 1992.

[8] Aure Anderson, F Ygge, "Managing Large Scale Computational Markets", http://www.enersearch. se/~ygge.

[9] B Hayes Roth, "A Blackboard Architecture for Control", Artificial Intelligence Journal, 26, pp 251-321, 1985.

[10] B Hayes Roth, "An Architecture for Adaptive Intelligent Systems", Artificial Intelligence, 72, pp 329-365, 1995.

[11] B J Grosz and C L Sidner, "Plans for Discourse", Intentions in Communication, (Eds P R Cohen, J M organ and M E Pollack), pp 417-444, MIT Press, 1990.

[12] Bernardo A Huberman, Tad Hogg, "Distributed Computation as an Economic System", Journal of Economic Perspectives, 9, 1, pp 141-152, 1995.

[13] C E Hewitt and W A Kornfield, "Message Passing Semantics", SIGART Newsletter, pp48, 1980.

[14] Carl A Waldspurger, Tad Hogg, Bernardo A Huberman. Jeffery O Kephart, W Scott Stornetta, "Spawn: A Distributed Computational Economy", IEEE Transactions on Software Engineering ,18,2, pp 103-117, Feb. 1992.

[15] Carl Hewitt and Jeff Inman, "DAI Betwixt and Between: From Intelligent Agents to Open Systems Science", IEEE Transactions on Systems, Man, and Cybernetics, 21,6, pp 1409-1419, 1991.

[16] Cheng Gu, T Ishida, "Analyzing the Social Behavior of Contract Net Protocol", LNAI, 1038 (Berlin: Springer), pp 116-127, 1996.

[17] Cristiano Castelfranchi, "Social Power: A Point Missed in Multi-Agent, DAI, and HCl" In Yves Demazeau & Jean Pierre Muller, editors, Decentralized AI, pp 49-62, Elsevier Science Publishers, 1990.

[18] Cristiano Castelfranchi, Maria Micelli, Amedo Cesta, "Dependence Relations Among Autonomous Agents", In Eric Werner & Yves Deamzeau, editors Decentralized AI, 3, pp 215-227, Elsevier Science Publishers, 1992.

[19] D Chapman, "Planning for Conjunctive Goals", Artificial Intelligence, 32, pp 333-378, 1987.

[20] D J Musliner, J A Hendler, A K Agrawala, E H Durfee, J K Strosnider, C J Paul, "The Challenges of Real-time AI", IEEE Computer, pp 58-66, Jan. 1995.

[21] D Partridge, "The Scope and Limitations of Future Generation Expert Systems", Future Generation Computer Systems, 3, 1, pp 1-10, 1987.

[22] D Reicken (Guest Editor), Communications of the ACM - Special Issue on Intelligent Agents, 37,7, 1994.

[23] Daniel D Corkill and Victor R Lesser, "The Use of Meta Level Control for Coordination in a Distributed Problem Solving Network", Proceedings of Eighth International Conference on Artificial Intelligence, August 1983, pp 748-756, (Also Published in Computer Architectures for Artificial Intelligence Applications, Benjamin W Wah and G J Li (Eds), IEEE Computer Sociey Press, pp 507-515, 1986.

[24] Daniel D Corkill, "Hierarchical Planning in a Distributed Environment", In Proceedings of Sixth International Joint Conference on Artificial Intelligence, pp 168-175, Cartridge, Massachusetts, August 1979

[25] Daniel E Atkins, William P Burmigham, Edmund H Durfee, Eric Glover, Tracy Mullen, Elke A Runmensteiner. Eliot Soloway, Jose M Vidal, Raven Wallace, Michael P Wellman. "Building the University of Michigan Digital

Library: Interacting Software Agents in Support of Inquiry-Based Education", IEEE Computer Special Issue on Building Large Scale Digital Libraries, May 1996.

[26] E H Durfee and Victor R Lesser, "Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation", IEEE Transactions on Systems, Man and Cybernetics, Special Issue on Distributed Sensor Networks, SMC-21,5, pp 1167-1183, September 1991.

[27] E H Durfee, "The Distributed Artificial Intelligence Melting Pot", IEEE Transactions on Systems, Man, and Cybernetics, 21,6, pp 1301-1306, 1991.

[28] E H Durfee, and Jeffery Rosenchien. "Distributed Problem Solving and Multi Agent Systems: Comparisons and Examples", In Proceedings of the Thirteenth International Distributed Artificial Intelligence Workshop, pp 94-104, July 1994.

[29] E H Durfee, D L Kiskis, W P Birmingham, "The Agent Architecture of University of Michigan Digital Library", IEEE Proc. of Software Engineering, 144(1), pp 61-71, 1997.

[30] E H Durfee, V R Lesser, "Using Partial Plans to Coordinate Distributed Problem Solvers", International Joint Conf. On Artificial Intelligence, IJCAI-87, pp 875-883, 1987.

[31] EH Durfee, V R Lesser, and D D Corkill, "Coherent Cooperation among Communicating Problem Solvers", IEEE Transactions on Computers, C,36,ll, pp 1275-1291, 1987.

[32] E H Durfee, V R Lesser, D D Corkill, "Cooperation through Communication in a Distributed Problem Solving Network", in Distributed Artificial Intelligence (Eds M N Huhns), pp 29-59, Pitman Publishing, 1988.

[33] E H Durfee, Victor R Lesser and Daniel D Corkill, "Trends in Distributed Problem Solving", IEEE Transactions in Knowledge and Data Engineering, KDE-1,1, pp 63-83, March 1989.

[34] E Sacerdoti, "Planning in a Hierarchy of Abstraction Spaces", Artificial Intelligence, 5, pp 115-135, 1974.

[35] E Sacerdoti, "The Nonlinear Nature of Plans", in Proc of Fourth International Joint Conf On Artificial Intelligence, IJCAI 75, pp 206-214, 1975.

[36] Edurdo Alonso. "How Individuals Negotiate Societies", IEEE Proc of ICMAS-98, pp 18-25. 1998

[37] F Ygge, H Akkermans, "On Resource Oriented Multi Commodity Market Computation", IEEE Proceedings of ICMAS-98, pp 365-371, 1998

[38] F Ygge, H Akkermans, "Power Load Management as a Computational Market", http://www.enersearch.se/~ygge.

[39] G Agha and C E Hewitt, "Concurrent Programming Using ACTORS: Expliotiing Large Scale Parallelism", A I Memo 865, MIT, 1985.

[40] G Agha, " ACTORS: A Model of Concurrent Computation in Distributed Systems", MIT Press, 1986.

[41] G F Couloris, J Dollimore and Kindberg, "Distributed Systems Concepts and Design", Addison Wesley Publishing Company, Reading, May, 1994.

[42] G Uma, B E Prasad, and Nalini Kumari, "Distributed Intelligent Systems : Issues Perspectives and Approaches", Knowledge Based Systems, 6, 2, pp 77-86, June 1993.

[43] G Uma, L O Alvares, Y Demazeau, "On Decomposition Methodology", Journal of IETE (Special Issue based on papers presented at the First National Conference on Parallel and Distributed AI), 42, 3, pp 111-116, 1996.

[44] G Uma, Luis Alvares and Yves Demazeau, "On Problem Decomposition in DAI", First Ibero-American Workshop on DAI and MAS, pp 25-26 Oct, 1996.

[45] G Wiederhold, P Wegner, S Cefi, "Toward Megaprogramming", Communications of the ACM, 33, 11, pp 89-99, 1992.

[46] G Zlotkin and J Rosenchein, " Cooperation arid Conflict Resolution via Negotiation among Autonomous Agents in Non Cooperative Domains", IEEE Transactions on Systems, Man, and Cybernetics (Special Issue on DAI), 21,6, pp 1317-1324, 1991.

[47] Gerhard Weiss (ed), "Multiagent Systems A Modern Approach to Distributed Artificial Intelligence", The MIT Press Cambridge London England, 1999.

[48] H J Levesque, P R Cohen and J H Nunes, "On Acting Together", Proc Ninth National Conference on AJ, pp 94-99, 1990.

[49] H Kautz, B Selman, and M Shah, "The hidden web", AI Magazine. 18,2, pp 27-35, 1997.

[50] H Van Dyke Parunak, "DAI and Manufacturing Control Some Issues and Insights", Decentralized AJ: Proc of First European Workshop on Modeling

Autonomous Agents in a Multi-Agent World Eds. Yves Demazeau and J P Muller, Elsevier, pp 81-101, 1989.

[51] Hirofumi Yamaki, Yukata Yamauchi, Tom Ishida, "Implementation Issues on Market Based QoS Control", IEEE Proc. ICMAS-98, pp357-364, 1998.

[52] I A Ferguson, "Towards an Architecture for Adaptive. Rational. Mobile Agents", In Decentralized AJ-3, Editors Eric Werner & Y Demazeau, pp249-262, Elsevier Science Publishers, 1992.

[53] J Ferber and P Carle, "Actors and Agents as Reflective Concurrent Objects: a Merging IV Perspective", IEEE Transactions on Systems, Man, and Cybernetics, 21, 6 pp 1420-1436, 1991.

[54] J Musliner, E H Durfee, K G Shin, "CIRCA: A cooperative intelligent real-time control architecture", IEEE Transactions on Systems, Man, and Cybernetics, 23, 6, pp!561-1574, Nov/Dec 1993.

[55] J P Muller, M Pischel, "Modelling Interacting Agents in Dynamic Environments", in Proc. of Eleventh European Conf. On Artificial Intelligence, ECAI-94, pp709-713, 1994.

[56] J P Muller, M Pischel, M Thiel, "Modelling Reactive Behavior in Vertically Layered Agent Architectures", in Wooldridge M and Jennings N R, editors, Intelligent Agents: Theories, Architectures and Languages, LNAI, vol. 890, pp 261-276, Springer Verlag, 1995.

[57] J P Searle, "Collective Intentions and Actions", in Intentions in Communication, (Eds P R Cohen, J M organ and M E Pollack), pp 401-416, MIT Press, 1990.

[58] J R Galliers, "A Strategic Framework for Multi Agent Cooperative Dialogue", Proceedings of the European Conference on Artificial Intelligence, pp 415-420, 1988.

[59] J S Rosenchein and M R Genesereth, "Deals Among Rational Agents", in Proc. of the Ninth International Conference on Artificial Intelligence (1JCA1-85), 91-99, 1985.

[60] J S Rosenchien, "Synchronization of Multi-Agent Plans", Proc 1st National Conference on AI, ppl 15-119, 1982.

[61] Jaime S Sichman, "DEPINT : Dependence-Based Coalition Formation in an Open Multi-Agent Scenario", Journal of Artificial Societies and Social Simulation, 1, 2, March 1998

[62] Jaime S Sichman, Y Demazeau, "Exploiting Social Reasoning to Enhance Adaption in Open Multi-agent Systems", in J Wainer & Carvalho, editors, Advances in AI, LNAI 991, pp 253-263, Springer Verlag (and ICMAS-95), 1995.

[63] Jaime Sichman, Yves Demazeau and Olivier Boissier. "When can Knowledge Based Systems be Called Agents[9]", In Proceedings of 9th Brazilian Symposium on Artificial Intelligence, (SBIA-92), Rio de Janeiro, Brazil, Setembro, 1992.

[64] Jaime Sichman and Yves Demazeau, "Exploiting Social Reasoning to Deal with Agency Level Inconsistency", In Proceedings of First International Conference on Multi Agent Systems, (ICMAS-95), San Fransisco, USA, June 1995.

[65] Jaime Sichman and Yves Demazeau, "Using Class Hierarchies to Implement Social Reasoning in Multi Agent Systems", In Proceedings of 11th Brazilian Symposium on Artificial Intelligence, (SBIA-94), Fortaleza, Brazil, Outubro, 1994.

[66] Jaime Sichman, R Conte, Y Demazeau, C Castelfranchi, "A Social Reasoning Mechanism Based on Dependence Networks", in Proc. of 11 European Conf. On AI, Amsterdam, pp 188-192, 1994.

[67] Jaime Simao Sichman, Yves Demazeau, "A Model for the Decision Phase of Autonomous Belief Revision in Open MAS", Journal of Brazilian Computer Society, 31, pp 40-50, 1996.

[68] James Kurose, Rahul Simha, "A Microeconomic Approach to Resource Allocation in Distributed Computer Systems", IEEE Transactions on Computers, 38, 5, pp 705-711, May 1989.

[69] Jon Doyle, "Rationality and its Roles in Reasoning", in Proc of Eighth National Conf. On AJ, pp 1093-1100, 1990.

[70] K P Sycara and D Zeng, "Coordination of Multiple Intelligent Software Agents", International Journal of Intelligent and Cooperative Information Systems, 5, 2&3, pp 181-211, 1996.

[71] K P Sycara, "Multiagent Systems", AI Magazine, Summer 1998, pp79-92, 1998.

[72] K P Sycara, "RETSINA - Reusable Environment for Task Structured Intelligent Network Agents", http://www.cs.cmueduA-softaeents/retsina/retsinahtml

[73] K P Sycara, K Decker, A Pannu, M Williamson and D Zeng, "Distributed Intelligent Agents", IEEE Expert, 11, 6, pp 36-46, December 1996.

[74] K S Decker, "Distributed Problem Solving Techniques: A Survey", IEEE Transactions on Systems, Man, and Cybernetics, 17, 5, pp 729-740, 1987.

[75] K S Decker, Alan Garvey, M A Humphrey, V R Lesser. "A Real-time Control Architecture for an Approximate Processing Blackboard System", International Journal of Pattern Recognition and AJ, 7, 2, pp 265-284, 1993.

[76] K S Decker, K Sycara and M Williamson, "Cloning for Intelligent Adaptive Information Agents", in C Zhang and D Lukose (eds), Multi Agent Systems, Springer Verlag, pp 63-75, 1997

[77] K S Decker, Katia Sycara, "Intelligent Adaptive Information Agents", Journal of Intelligent Information Systems, 9, pp 239-260, 1997.

[78] K S Decker, M Williamson, K Sycara, "Modeling Information Agents: Advertisements, Organizational Roles and Dynamic Behavior", in Proc. AAAI 96, Workshop on Agent Modeling, Report WS-96-02, 1996.

[79] L D Erman and V R Lesser, "A Multi Level Organization for Problem Solving Using Many Diverse Cooperating Sources of Knowledge", Proc. of International Conference on AI, pp 483-490, 1975.

[80] Les Gaser, "An Overview of DAI", in Distributed Artificial Intelligence: Theory and Praxis, Eds. Nicholas M Avouris and Les Gasser, Kluwer Academic Publishers, 1992.

[81] Luc Steels, "Cooperation Between Distributed Agents Through Self Organization", in Demazeau Y, Muller JP, editors, Decentralized AI, Proc of First European Workshop on Modeling Autonomous Agents in Multi Agent World (MAAMAW-89), pp 175-196, Elsevier Science Publishers, 1990

[82] M E Bratman, D J Isreal, M E Pollack. "Plans and Resource Bounded Practical Reasoning", Computational Intelligence, 4, pp 349-356, 1987

[83] M E Pollack, "The Use of Plans", AI, 57, 1, pp 43-68, 1992

[84] M E Pollack, M Riguette, "Introducing The Tile World: Experimentally Evaluating Agent Architecture", Proc of Eighth National Conference on AI, AAAI-90, Boston MA, pp 183-189, 1990

[85] M P Georgeff, A L Lansky. "Reactive Reasoning and Planning". Proc of Sixth National Conference on AI. AAAI-87, Seattle WA, pp 268-272, 1987

[86| M P Singh, "Group Intentions", Proc of Tenth International Workshop on Distributed Artificial Intelligence, Texas, MCC Technical Report ACT-AI-355-39, 1990.

[87] M R Genesereth, S P Ketchpel, "Software Agents", Communications of the ACM, 37,7, pp 48-53, 1994.

[88] Marie-Pierre Gervais, "Enhancing Telecommunications Service Engineering with Mobile Agent Technology and Formal Methods", IEEE Communications Magazine, pp 38-43, July 1998.

[89] Michael P Wellman, "A Market Oriented Programming Environment and its Application to Distributed Multi Commodity Flow Problems", Journal of AI Research, August 1993, pp 1-23, 1993.

[90] Michael P Wellman, "The Economic Approach to Artificial Intelligence", ACM Computing Surveys, 28 (4es), December 1996, http://ai.eecs.umich.edu/people/wellman/wellman.html.

[91] Michael P Wellman, W P Birmingham, E H Durfee, "The Digital Library as a Community of Information Agents", IEEE Expert, 11(3), pp10-11, 1996.

[92] Michael R Genesereth, Matthew L Ginsberg, Jeffery S Rosenchien, "Cooperation without Communication", In Proc. of the 11th National Conf. On AI, pp 51-57, Philadelphia, Aug 86.

[93] Michael S Greenberg and Jennifer C Byington, "Mobile Agents and Security", IEEE Communications Magazine, pp 76-85, July 1998.

[94] Michael Wooldridge, Nick Jennings, " Intelligent Agents: Theory and Practice", Knowledge Engineering Review, 10, 2, pp 115-152, 1995.

[95] Mun인der P Singh, "Multiagent Systems: A Theoretical Framework for Intentions, Know-How, and Communications", 799, Springer Verlag.

[96] N R Jennings, "Joint Intentions as a Model of Multi-Agent Cooperation in Complex Dynamic Environments", Ph.D. Thesis, Dept. of Electronic Engineering, Queen Mary and Westfield College, University of London, 1992.

[97] N R Jennings, Katia Sycara, Micheal Wooldridge, "A Roadmap of Agent Research and Development", Autonomous Agents and Multi Agent Systems, 1, DO 275-306. 1998.

[98] Nalini Kumari, G Uma and B E Prasad, "Sub-problem Allocation in Distributed Production Systems", Journal of IETE, 42, 3, pp 111-116, May-June 1996.

[99] O Shehory, K Sycara , P Chalasani, S Jha , "Agent Cloning: An Approach to Agent Mobility and Resource Allocation", IEEE Communications 36, 7, pp 58-67, July 1998.

[100] P Meas, "Agents That Reduce Work and Information Overload", Communications of the ACM, 37,7, pp 31-40, July 1994.

[101] PR Cohen and H J Levesque, "Intention is Choice with Commitment", Artificial Intelligence, 42, pp 213-261, 1990.

[102] PR Cohen and H J Levesque, "Teamwork", SRI Technical Note 504, 1991.

[103] Patrick Henry Winston, Artificial Intelligence, Addison Wesley, 1992.

[104] R A Brooks, "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, 2,1, pp 14-23, 1986.

[105] R A Brooks, "Intelligence without Reason", in Proc. of the Twelfth International Joint Conf. An Artificial Intelligence, IJCA1 91, pp569-595, Sydney, Australia, 1990.

[106] R A Brooks, "Intelligence without Representation", Artificial Intelligence, 47,ppl39-159, 1991.

[107] R Conte, M Miceli and C Castelfranchi, "Limits and Levels of Cooperation: Disentangling Various Types of Pro-social Interaction", Proc. Modelling An Autonomous Agent in Multi-Agent World, 1990.

[108] R Davis, and R G Smith, "Negotiation as a Metaphor for Distributed Problem Solving", Artificial Inelligence ,20,1, pp63-109, 1983

[109] R E Fikes, N Nilson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem-solving", Artificial Intelligence, 5,2, pp 189-208, 1971

[110] R Englemore and T Morgan (eds), "Blackboard Systems", Addison Wesley, 1988.

[111] R G Smith, "Contract Net Protocol High Level Communication and Control in Distributed Problem Solvers", IEEE Transactions on Computers, 29,12, pp 1104-1113, 1980.

[112] R G Smith, R Davis, "Frameworks for Cooperation in Distributed Problem Solvers", IEEE Transactions on Systems, Man, and Cybernetics ,11,1, pp 61-70, 1981.

[113] R James Firby, "An Investigation into Reactive Planning in Complex Domains", AAA1-87, pp202-206, f987.

[114] R P Bonasso, D Kortenkamp. D P Miller, and M Slack, "Experiences with an Architecture for Intelligent Reactive Agents", in Intelligent Agents II eds M Wooldridge, J P Muller, and M Tambe, pp 187-202, Lecture Notes in Artificial Intelligence 1037, Springer Verlag, 1996.

[115] R Steeb, S Cammaratta, F Hayes Roth, P Thorndyke, and R Wesson, "Distributed Intelligence for Air Fleet Control", in Readings in DAI Eds. A H Bond and L Gasser, pp 90-101, 1988.

[116] R Toumela and K Miller, "We Intentions", Philosophical Studies 53, pp 367-389, 1988.

[117] Richard D Fennel and Victor R Lesser, "Parallelism in Artificial Intelligence Problem Solving: A Case Study of Hearsay-H", IEEE Transactions on Computers C-26,2, pp 98-111, 1977.

[118] S E Conry , R A Meyer, and V R Lesser, "Multistage Negotiation in Distributed Planning" in Readings in DAI, Eds. A H Bond and L Gasser. pp 367-384, 1988.

[119] Sandip Sen and Edmund H Durfee, 'The Role of Commitment in Cooperative Negotiation", International Journal of Intelligent and Cooperative Information Systems, 3,1, pp 67-81, 1994.

[120] Sarit Kraus, Jeffery Rosenchein, "The Role of Representation in Interaction: Discovering Focal Points Among Alternate Solutuins", In Eric Werner, Y Demazeau, editors, Decentralized AI, 3, pp 147-165, Elsevier Science Publishers 1992.

[121] Shaheen Fatima, G Uma, "An Adaptive Organizational Policy for Multi Agent Systems -- AASMANT. IEEE Proceedings of the Third International Conference on Multi Agent Systems. Paris, ppl2u-127, July 1998

[122] Shaheen Fatima. G Uma, "Task Sharing in Multi Agent Systems A Study", at the National Symposium on Recent Trends in Information Technology, at P S G College of Technology, Coimbatore, 1997.

[123] Shaheen Fatima, G Uma, "AASMAn: An Adaptive Organizational Policy For A Society of Market Based Agents", 23, 4, Sadhana Academy Proceedings in Engineering Science Published by Indian Academy of Sciences, Bangalore, pp 377-392,Aug.1998.

[124] Shlomo Zilberstein, Stuart Russel, "Optimal Composition of Real-time Systems", 82, 2, pp 181-213, May 1996.

[125] Stephanie Cammarata, David Mc Arthur, Randall Steeb, "Strategies for Cooperation in Distributed Problem Solving", Proc. of Eighth International Conference on AI, August, pp 767-770, 1983.

[126] Stuart J Russel, Peter Norvig, "Artificial Intelligence - A Modern Approach", Prentice Hall International Inc, 1995.

[127] Stuart Russel, Eric Wefald, "Principles of Metareasoning", Artificial Intelligence, 49, pp 361-395, Elsevier Science Publishers, 1991.

[128] Stuart Russel, Shlomo Zilberstein, "Composing Real-time systems", Twelfth Int. Joint Conf. On Artificial Itelligence, IJCAJ 91, pp 212-217, 1991.

[129] T Dean, M Boddy, "An Analysis of Time Dependent Planning", in Proc. of Seventh National Conf. On AI, pp 49-54 1988.

[130] T Finin, R Fritzson, D Me Kay, R Me Entire, "KQML as an agent communication language", in Proc. of Third International Conference on Information and Knowledge Management, CIKM' 94, ACM Press, November 1994.

[131] T W Malone, R E Fikes, K R Grant, M T Howard, "Enterprise: A Market-like Task Scheduler for Distributed Computing Environments", The Ecology of Computation, North Holland, pp 177-205,1988

[132] T W Sandholm, "An Implementation of the Contract Net Protocol based on Marginal Cost Calculations", AAA1-93, pp 256-262, 1993

[133] T W Sandholm, V R Lesser, "Advantages, of Leveled Commitment Contracting Protocol", Tech Report 95-72, Univ. of Mass, Sep 1995

[134] Tad Hogg and Bernardo A Huberman. "Controlling Chaos in Distributed Systems", IEEE Transactions on Systems, Man, and Cybernetics, 21,6, pp 1325-1332, 1991.

[135] Timothy Finin, Yannis Labrou, Yuri Peng, "Mobile Agents can Benefit from Standards Efforts on Interagent Communication", IEEE Communications Magazine, pp 50-56, July 1998.

[136] Toru Ishida, "Parallel, Distributed, and Multi-Agent Production Systems", Proc. of the First International Conference on Multi-Agent Systems ICMAS-85, pp 416-422 (and LNAI 878), 1995.

[137] Toru Ishida, Les Gasser, Makoto Yokoo, "Organization Self-Design of Production Systems", IEEE Transactions on Knowledge and Data Engineering, 4,2, pp 123-134, 1992.

[138] Toru Ishida, Yutaka Sasaki, Keiko Nakata and Yoshimi Fukahara, "A Meta Level Control Architecture for Production Systems", IEEE Transactions on Knowledge and Data Engineering, 7, 1, pp 44-52, 1995.

[139] Tracy Mullen, Michael Wellman, "Some Issues on the Design of Market Oriented Agents", In M Wooldridge, J P Mullen, M Tambe (Eds), Intelligents Agents II - Agent Theories, Architectures and Languages. IJCAI-95 Workshop (ATAL), Montreal, Canada, Proceedings, Springer Verlag LNAI 1037, pp 283-289, Aug. 1995.

[140] V R Lesser, D D Corkill, "Functionally Accurate Cooperative Distributed Systems" IEEE Transactions on Systems, Man, and Cybernetics, 11,1, pp 81-96, 1981.

[141] V R Lesser, J Pavlin, E Durfee, "Approximate Processing in Real-time Problem Solving", AI Magazine, 9,1, pp 49-61, Spring 1988

[142] Victor R Lesser and Daniel D Corkill, "Distributed Problem Solving", In Stuart C Shapiro (editor), Encyclopedia of Artificial Intelligence, 1, pp 245-251, John Wiley and Sons, 1987.

[143] Victor R Lesser and Daniel D Corkill, "The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks", AI Magazine 4(3). Fall 1983. pp 15-33 (Also in Blackboard Systems, R E nglemore and T Morgan (Eds.), Addison-Wesley, 1988, pp 353-386, and Readings from the AI Magazine, Vols 1-5, 1980-1985, R Engelmore (ed) AAA1 Publishers, pp 69-85, 1988.

[144] Victor R Lesser and Lee D Erman, "Distributed Interpretation A Model and Experiment, IEEE Transactions on Computers C-29. 12. pp 1144-1163, 1980

[145] Victor R Lessor et al, "A High Level Simulation Testbed for Cooperative Distributed Problem Soling", Third International Conference on Distributed Computer Systems, pp 341-349, Oct 1983.

[146] Vu Ann Pham and Ahmed Karmouch, "Mobile Software Agents: An Overview", IEEE Communications Magazine, pp 26-37, July 1998.

[147] W Walsh, M Wellman, "A Market Protocol for Decentralized Task Allocation", IEEE Proc. ICMAS-98, pp 325-332, 1998.

[148] Wilmer Cripe, George Cebenko, Katsuhiro Moizumi, Robert Gray, "Network Awareness and Mobile Agent Systems", IEEE Communications Magazine, pp 44-49, July 1998.