Sophisticated Negotiation In Multi-Agent Systems

Xiaoqin Zhang*

UMass Computer Science Technical Report 2002-40

October 28, 2002

Abstract

As multi-agent systems are involved in more and more complicated applications, negotiation, a technique that can be used effectively to coordinate the behavior of agents, becomes increasingly important and difficult.

This dissertation addresses a number of problems for sophisticated negotiation in multi-agent systems. The agents are large-grained and complex with multiple goals and tasks. The organizational relationships among agents are heterogeneous, and the agents negotiate over multiple topics. Finally, the negotiation process is tightly interleaved with agents' controlling, scheduling and planning processes.

I identify the following problems: negotiation over multiple inter-related issues; negotiation in a complex organizational context; negotiation as a distributed search process. I then attack these problems from several directions. In negotiation decision process, I use the motivational quantities (MQ) framework to support reasoning based on organizational concerns; I build a partial order scheduler that allows the agent to evaluate the flexibility and the feasibility of multi-linked negotiation issues. I also introduce the multi-leveled negotiation framework, which allows the negotiation to be performed at different abstraction levels. I then introduce multiple negotiation topics which allow the agents negotiate in a multi-dimensional space.

I implement these approaches and validate them through experimental work. I show that efficient negotiation techniques improve agents' performance significantly. I also outline directions for improving and extending our approaches.

*Computer Science Department, University of Massachusetts, Amherst, MA 01003, Email: xqzhang@cs.umass.edu

SOPHISTICATED NEGOTIATION IN MULTI-AGENT SYSTEMS

A Dissertation Presented

by

XIAOQIN ZHANG

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2002

Department of Computer Science

SOPHISTICATED NEGOTIATION IN MULTI-AGENT SYSTEMS

A Dissertation Presented

by

XIAOQIN ZHANG

Approved as to style and content by:

Victor R. Lesser, Chair

Abhijit Deshmukh, Member

David Jensen, Member

Tom Wagner, Member

Shlomo Zilberstein, Member

W. Bruce Croft, Department Chair
Department of Computer Science

This dissertation is dedicated to my mother Dan Ou-Yang, and my father, Ming-Sheng Zhang.

ACKNOWLEDGMENTS

First of all, I would like give my sincere gratitude to Professor Victor Lesser, for his lots of intellectual stimulations and helpful discussions in this dissertation work, and also for his enduring caring, persistent support and warm encouragement. He is not only an admirable academic adviser but also a great mentor with a heart full of love.

I would also like to thank the help and support of my committee members. Professor Shlomo Zilberstein's reference of other related work helps me to think how my work can contribute to other problems. Professor David Jensen has helped me on analysis of experimental data. Professor Abhijit Deshmukh's attention to detail helps me clarify my ideas and writing. Dr. Tom Wagner has helped a lot in using and understanding both the MQ framework and the DTC scheduler.

I would also like to acknowledge each member in MAS lab, Ping Xuan, Bryan Horling, Anita Raja, John Aseltine, Roger Mailler, Jiaying Shen, Mark Sims, Sherief Abdallah, Kyle Rawlins, and Claudia Goldman. They provide a warm and friendly environment which makes my six-year graduate student life seems not that long. Specially, I want to thank Bryan Horling's always-in-time technical support on JAF framework and MASS simulator, which makes the experiments in this thesis possible. I also like to appreciate Anita Raja's friendship which supports me from different perspectives. Additionally, I would like to thank Mark Sims and Michele Roberts for their help on editing this thesis.

Certainly, I would like to acknowledge the love and support from my family and friends which helped me through this long journey. My mother has put her incessant love and care on each step I walked and will be with me always. My father started to educate me with mathematics and literature when I was three years old, and his encouragement and support never leave me. I feel very luck to have a sister, Xiaoyan Zhang, who shares the four seasons with me. My patient and loving husband, Liang Li, deserves my hearty appreciation, for his all kinds of support including the technical support for my working environment at home, and for his sharing of all my suffering in this process. My friends Wenqi You and Xiaoyan Li, also deserve my special thanks, for their friendship and support in these recent years.

Last but not least, I would like to give thanks to God, whose love and guide led me through this process. Those intellectual contributions listed in this work are just next to nothing in front of him, but his love to me is immeasurable.

ABSTRACT

SOPHISTICATED NEGOTIATION IN MULTI-AGENT SYSTEMS

SEPTEMBER 2002

XIAOQIN ZHANG

B.S., UNIVERSITY OF SCIENCE & TECHNOLOGY OF CHINA

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Victor R. Lesser

As multi-agent systems are involved in more and more complicated applications, negotiation, a technique that can be used effectively to coordinate the behavior of agents, becomes increasingly important and difficult.

This dissertation addresses a number of problems for sophisticated negotiation in multi-agent systems. The agents are large-grained and complex with multiple goals and tasks. The organizational relationships among agents are heterogeneous, and the agents negotiate over multiple topics. Finally, the negotiation process is tightly interleaved with agents' controlling, scheduling and planning processes.

I identify the following problems: negotiation over multiple inter-related issues; negotiation in a complex organizational context; negotiation as a distributed search process. I then attack these problems from several directions. In negotiation decision process, I use the motivational quantities (MQ) framework to support reasoning based on organizational concerns; I build a partial order scheduler that allows the agent to evaluate the flexibility and the feasibility of multi-linked negotiation issues. I also introduce the multi-leveled negotiation framework, which allows the negotiation to be performed at different abstraction levels. I then introduce multiple negotiation topics which allow the agents negotiate in a multi-dimensional space.

I implement these approaches and validate them through experimental work. I show that efficient negotiation techniques improve agents' performance significantly. I also outline directions for improving and extending our approaches.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

OVERVIEW

In Section 1.1 we first introduce the research area of multi-agent systems, and in Section 1.2 we briefly review existing research on negotiation. Given that background, in Section 1.4 we discuss the major research questions addressed in this thesis and the motivation for these questions. We also provide a high-level overview of the approaches and ideas used to solve them. The key assumptions we have made to solve these questions are identified in Section 1.3, and the major research results and contributions are presented in Section 1.5. Section 1.6 provides a road map for the rest of this thesis.

1.1 Multi-Agent Systems and Software Agents

Multi-agent systems (MAS) have become increasingly important over the last ten years as the Internet has become a reality. Computers are no longer stand-alone devices; they are tightly connected to each other and their users. Current computer systems are large, open and distributed, and include heterogeneous computer platforms. Modern applications such as distributed data processing, intelligent environment controlling, distributed sensor networks, and e-commence, involve a number of computers and/or large amounts of data, which are distributed geographically and/or belong to different owners. The centralized computing model is not suitable for these distributed applications. Furthermore, a centralized approach also lacks the capability to handle the complexity of these applications, in which computers act more like individual agents rather than single parts. Computers need more autonomy to react to dynamic, open environments that contain uncertainty. Multi-agent systems technology is needed to handle the high-level interactions among these individual computer systems.

A multi-agent system is built out of software agents, which are autonomous, intelligent, computational entities that pursue goals or carry out tasks in order to meet their design objectives. Agents are:

1. *Computational*: They physically exists in the form of programs that run on computing devices [1].

2. *Autonomous*: To some extent they have control over their behavior and can act without the intervention of human or other systems.

[1] This definition does not apply to human agents.

3. *Intelligent*: To say that agents are intelligent does not mean that they know everything or that they are capable of any task, nor does it mean that they never fail; rather, it means that they operate flexibly and rationally in a variety of environmental circumstances, given the information they have and their perceptual and effectual capabilities, such that they optimize some given performance measures.

In multi-agent systems, agents interact, meaning that they may be affected by other agents or perhaps by humans when they are pursuing their goals and executing their tasks. Interaction can take place directly through a shared language or indirectly through the environment in which they are embedded. The interaction in multi-agent systems can be used for coordination among agents over a common goal or a shared task, either in cooperative or competitive situations. The long-term goal of multi-agent systems research is to develop mechanisms and methods that enable agents to interact as well as humans (or even better), and to understand interaction among intelligent entities whether they are computational, human, or both.

Furthermore, multi-agent systems are capable of playing an important role in the studying of theories related to human societies and organizations [68]. There are many similar aspects between multi-agent systems and human societies: they are constructed of intelligent individuals; there are different relationships among these individuals; each individual has only limited knowledge and is resource-bounded; the individuals interact with each other in that they coordinate, negotiate, share knowledge, transfer information, and form and dissolve all kinds of organizations and groups. Although the interactive processes among humans happen everyday and everywhere around us, they are not fully understood yet. The experimental study of human societies and organizations is difficult, because it is very hard to manipulate human beings to set up the experimental environment. Also, the data collection process not only is time consuming but also is impossible sometimes given that some people are not willing to provide the data. Based on their similarity to human societies, multi-agent systems can be used to simulate human organizations and societies, to develop, test, and verify the models and theories of interactivity in human societies.

1.2 Negotiation in Multi-Agent Systems

Negotiation is the interactive communication among agents to facilitate a distributed search process. Therefore, it is an important technique that can be used to effectively coordinate the behavior of agents in multi-agent systems. Negotiation is used for the allocation of tasks and resources, the recognition and resolution of conflicts, the determination of organizational structures, and, hence, for the coherence of the agent society.

As Muller [37] suggests, the research on negotiation can be classified into three categories: *negotiation language*, *negotiation decision* and *negotiation process*.

1. The *negotiation language* category concerns the inter-agent communication part of negotiation such as the primitives for negotiation, their semantics and their usage in terms of a negotiation protocol. This category also includes the exploration of the structure of the negotiation topics.

Most work on *negotiation language primitives* are built on speech acts theory [4, 55]. The negotiation primitives used by agents can be grouped as: *Initiators* (such as Propose, Arrange, Request, Inform, Query, Command, etc.), *Reactors* (such as Answer, Refine, Modify, Change, Bid, Reply, etc.), and *Completers* (such as Confirm, Promise, Commit, Accept, Reject, Agree, etc.). These primitives are taken from several research projects [10, 24, 27, 46, 20, 69].

A special context is transmitted together with a negotiation language primitive. The *negotiation object structure* represents what will be negotiated over; we will call this the "negotiation topic". This may be plans [27, 64], time intervals for possible appointments [46], offers or tasks [20], or costs and prices. The negotiation topic is involved in the negotiation decision process, since the agent uses it in its local decision-making process on negotiation. Therefore, selection of the negotiation topic influences the negotiation performance.

Negotiation protocols define the possible sequence of negotiation actions. Usually the protocols are defined as finite automata [11]. The Contract Net Protocol (CNP) [60] established a basic negotiation protocol model that involves a process of task announcing, bidding evaluation, and contract awarding. There are many variations of this basic protocol. For example, leveled commitments [47, 48] allow an agent to decommit from a contract by paying a decommitment penalty.

2. The *negotiation decision* category deals with the intra-agent part of negotiation, such as the evaluation process for selecting bids and algorithms used to compare the negotiation topics. The definition of utility functions and the representation and structure of the agents' preferences also belong to this category, as do negotiation strategies.

   Utility functions have been studied extensively in game-theory approaches to multi-agent systems [72, 74, 73, 43]. The utilities are given as decision matrices where an agent can look up a value for a certain action. The negotiation decisions are based on the optimization of utility.

   When negotiation is used for conflict resolution and task sharing, the negotiation decisions are based on complex negotiation objects like plans. In this sense, it is important to develop *comparing and matching functions* for complex negotiation objects. Kakehi and Tokoro use plans (a sequence of actions) as negotiation objects; the matching function thus computes the set of maximal non-conflicting sequence of actions [27].

   Negotiation strategies describe the agents' attitude in the negotiation process: they may concede unilaterally (not benefiting any participant), act competitively or act cooperatively [42]. Additionally, negotiation strategies decide the tactics to be used to resolve conflicts such as solving them in sequence, sorting them according to their importance, taking the least different one, or reducing all criteria to one.

3. The *negotiation process* category is concerned with the global negotiation behavior of the individual agent and the negotiating society. General models of the negotiation process are investigated and the global behavior of the negotiation participants are analyzed in this category.

The procedural negotiation model defines the behavior of an agent during the negotiation process. Bussmann and Muller [9] present a negotiation algorithm as a two-cycle process model: the pure negotiation cycle and the conflict resolution cycle which run in parallel. Progressive negotiation [33, 32] is proposed to deal with negotiation that involves multiple homogeneous agents. The negotiation among a group of agents can be divided into a number of sub-negotiations which proceed incrementally.

System behavior and analysis work analyzes the behavior of the negotiation society and considers the quality of the problem-solving process in the context of negotiation versus centralized processing. Burkhard [8] has done work on the analysis of properties of multi-agent systems, concerning liveness and fairness for different kinds of agent societies. Kosoresow [28] generated complexity results in analyzing the algorithms which implement the agents' interactions. He showed that for a special case, the total amount of work to be done in a multi-agent system with negotiation is less than or equal to the total work in centralized coordination systems.

1.3 Basic Assumptions

The following assumptions are adopted in this thesis:

1. Agents are autonomous, intelligent, computational entities. Agents are also rationally and resource bounded, and have limited knowledge of the environment and the state of other agents.

2. Agents are utility-driven and designed to maximize their local utility given their limited resources. An agent may have multiple user-defined organizational goals. An agent's overall utility depends on the progress towards its goals.

3. Agents have multiple tasks related to their goals. The tasks are complex, and there are several alternative ways to perform them. Agents have the ability to choose which tasks to perform, when to perform them, and how to perform them.

4. There are soft real-time concerns. Each task has an earliest start time and a deadline constraint. A tasks needs to be performed between its earliest start time and its deadline to produce valid results. Usually, finishing a task before the deadline brings an agent a higher reward.

5. There is a decommitment penalty associated with each commitment. Once two agents have confirmed a commitment, the agent that wants to decommit from its commitment needs to pay a previously agreed upon decommitment penalty to the other agent.

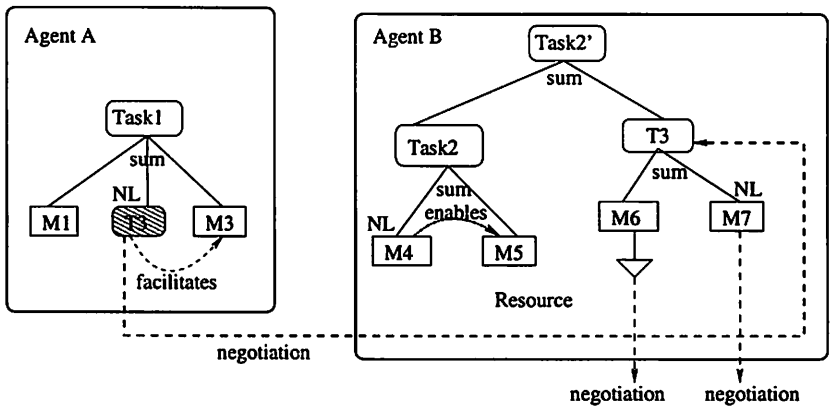


Figure 1.1: Multi-linked negotiation: example 1

1.4 Motivation and Major Ideas

This thesis research is motivated by the following three questions: how agents deal with multiple related negotiation issues; how agents negotiate in a complex organizational context, and how agents perform efficient negotiation as a distributed search process.

1.4.1 Multi-linked Negotiation

The first question concerns how an agent deals with multiple (possibly concurrent) negotiation issues when these issues are interconnected. In a multi-task, resource sharing environment, an agent needs to deal with multiple, related negotiation issues including:

1. tasks contracted to other agents;
2. tasks requested by other agents;
3. external resource requirements for local activities;
4. interrelationships among activities distributed among different agents.

Here are some examples. Figure 1.1 shows a situation where agent A has a nonlocal task $T3$ that it wants to contract out to agent B. If agent B takes task $T3$, it needs to subcontract $M7$ (a subtask of $T3$) to another agent and request a resource for $M6$ (another subtask for $T3$) through negotiation. Agent B has three inter-related negotiation issues to handle: the negotiation with agent A on task $T3$, the negotiation on $M7$, and the negotiation on the resource for $M6$. These three negotiation issues are related to each other. When and how $M7$ is performed, and when and for how long the resource is available for $M6$, affect when and how task $T3$ is performed.

Figure 1.2 shows another situation where agent A has a nonlocal task $M2$ that it wants to contract out to agent B while agent B has a nonlocal task $M4$ contracted to agent C. The negotiation on $M2$ is related to the negotiation on $M4$, because when $M4$ is performed

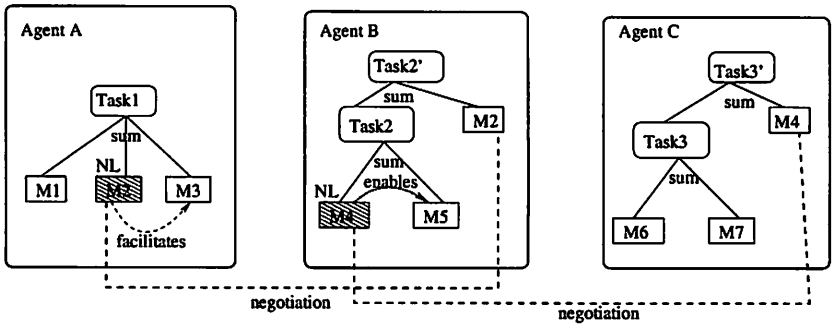


Figure 1.2: Multi-linked negotiation: example 2

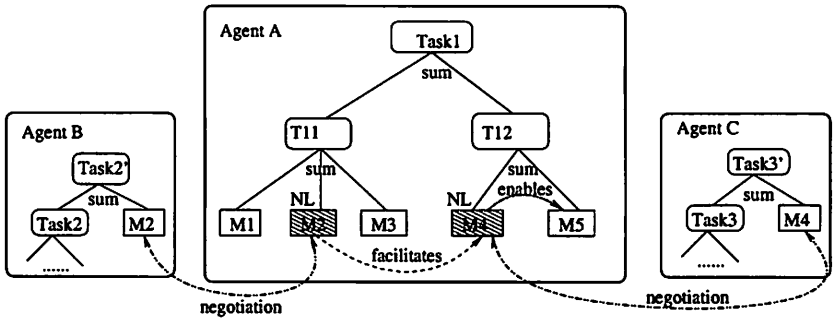


Figure 1.3: Multi-linked negotiation: example 3

affects when $M5$ can be started[2]. Given the limited processing capability of agent B that only one task can be performed at a time, therefore, when $M5$ can be performed influences when $M2$ can be performed.

Figure 1.3 shows a final example where agent A has two nonlocal tasks: $M2$ needs to be contracted out to agent B and M4 needs to be contracted out to agent C. The negotiation on $M2$ and the negotiation on $M4$ are related because of the "facilitates" relationship from $M2$ to $M4$. If $M2$ is finished before $M4$ starts, it can facilitate $M4$ by reducing its processing time or cost, or improving its quality achievement.

In general, multi-linked negotiation describes a situation where one agent needs to negotiate with multiple other agents about different issues, and where the negotiation over one issue has influence on the negotiations over other issues. The commitment on one issue affects the evaluation of a commitment or the construction of a proposal for another issue.

How can the agent deal with these multiple related negotiation issues? One approach is to deal with these issues independently just like separate issues, ignoring their interactions[3]. If these negotiations are performed concurrently, there could be possible conflicts among the solutions to these negotiation issues; hence the agent may not be able to find a combined feasible solution that satisfies all issues without re-negotiation over some already "settled"

[2]The "enables" relationship from $M4$ to $M5$ specifies that $M5$ can only start after $M4$ is finished.

[3]This approach seems too naive, but is commonly used currently. Most research only deals with single issue negotiation; no work has been done to study the relationships among different negotiation issues.

issues. For example, in Figure 1.1, agent B negotiates with agent A and promises to finish $T3$ by time 20. Meanwhile, agent B also negotiates with another agent about $M7$ and gets a contract that $M7$ will be finished at time 30. Agent B then finds it is impossible for $T3$ be finished by time 20 given that its subtask $M7$ will not be finished until time 30.

To reduce the likelihood that this type of conflict occurs, these negotiations could be performed sequentially; the agent deals with only one negotiation issue at a time, and later negotiations are based on the results of earlier negotiations. This sequential process, however, is not a panacea. First of all, the negotiation process takes much longer when all the issues need to be negotiated sequentially, potentially using up valuable time (delaying when problem solving can begin) and reducing the potential solution space. For example, in Figure 1.1, suppose the deadline for completion of $M7$ is 20, the same as $T3$. If the negotiation on $M7$ starts at 10 and finishes at time 12, then $M7$ can only start after time 12. If the negotiation on $M7$ starts at time 3, there is a larger time slot for the execution of $M7$; hence, it is easier for the negotiation on $M7$ to succeed. Additionally, when the negotiation deadline is taken into consideration, a negotiation started late may lose any chance of success. For instance, in Figure 1.1, suppose agent A associates a negotiation deadline of 8 with the proposal of $T3$. If agent B replies to this proposal later than time 8 because it wants to settle its other negotiation issues first, it can not get the contract on $T3$ accepted.

Secondly, even if all the issues are negotiated sequentially, there is no guarantee of an optimal solution or even of any possible solution. This problem can occur if the agent does not reason about the ordering of the negotiation issues and just treats them as independent issues, with their ordering being random. In this situation, the results from the previous negotiations may make later negotiation issues very difficult or even impossible to succeed. For instance, in Figure 1.1, if agent B first negotiates about $T3$ before starting the negotiations on $M6$ and $M7$, and the promised finish time of $T3$ results in tight constraints on the resource request of $M6$ and the negotiation on $M7$, these negotiations may fail and the commitment on $T3$ would have to be decommitted from.

One additional problem is caused by the difficulty in evaluating a commitment given that later issues are undecided, and making it harder for an agent to find a local solution that will contribute effectively to the construction of a good global solution. For example, in Figure 1.3, agent A has two non-local tasks, task $M2$ contracted to agent B and task $M4$ contracted to agent C. If $M2$ could be finished before M4 starts, it would reduce the processing time of $M4$ by 50%. Suppose agent B first negotiates with agent C and then negotiates with agent A; through the negotiation with agent C, it is decided that $M4$ starts at time 20 and finishes by time 40, but then it is found that task $M2$ could finish at time 25. Given this later information, if the start of $M4$ is delayed to time 25, $M4$ actually could be finished at time 35 because of the *facilitates* effect. This solution would not be found, however, if the agent ignores the interactions among these negotiation issues.

These previous examples indicate how important it is for an agent to reason about the interactions among different negotiation issues and manage them from a more global perspective. If done effectively, this permits the agent to minimize the possibility of conflicts among the different negotiation issues and then achieve better performance.

Additionally, these examples show that it is difficult to deal with multi-linked negotiation problems because:

1. There are possible conflicts among related issues. If not resolved, these conflicts may cause the agent's local plan to fail or reduce its local utility achievement.

2. There are uncertainties associated with negotiations. Since the agent does not have perfect and complete knowledge of the environment and the other agents' states, the result of a negotiation is uncertain. The agent may have an estimation about the negotiation outcome, but it needs to be prepared for different outcomes.

3. There is a cost for negotiation. On one hand, the agent needs to allocate valuable computational and communication resources for negotiation. On the other hand, the time spent on negotiation may affect the outcome of the negotiation. For example, the longer time is spent on negotiation may reduce the time available for execution hence reducing the possibility of finding a solution. Similarly, a delayed reply to a contract may not be accepted if other agents that replied to it earlier.

4. The negotiation process is interleaved with the agent's local planning and scheduling process because the agent needs to find a feasible local solution that satisfies all commitments and local constraints.

The multi-linked negotiation problem is not only a complicated problem, but also an important one because it actually happens in a number of application domains. For example, in a supply chain problem, negotiations go on among more than two agents. The consumer agent negotiates with the producer agent, and the producer agent needs to negotiate with the supplier agent. The negotiation between the producer agent and the supplier agent has a direct influence on the negotiation between the producer agent and the consumer agent. Figure 1.4 shows a supply chain example [4]. There are a number of companies, some of which produce parts for computers and some of which assemble computers; also others are distributors, stores and customers. Multi-linked problems occur throughout this system. Another example of multi-linked negotiation is the distributed sensor network application [23]. There are multiple sensors distributed at different locations, each of which has different coverage. Multiple targets move through the region and it takes a certain number (more than one) of sensors to track a target so as to get good tracking quality. Which sensors should be used to track which target during which time interval poses an interesting multi-linked negotiation problem too.

To our knowledge, there is no work that has explicitly addressed the multi-linked negotiation problem. Some work, however, does look at agents that deal with multiple negotiation issues concurrently. Leveled commitment [48] allows agent to decommit by paying a decommitment penalty. A statistical model is used to predict future events so that the agent can calculate the opportunity cost for the current commitment. When a new task comes, the agent can backtrack from its previous decision by paying a decommitment penalty to get a better local solution. In the distributed meeting scheduling problem [56, 59], multiple meeting scheduling processes are going on concurrently. The agent can either block the proposed time or not block it until an agreement is reached; the commitment strategy affects the system's performance. An adaptive selection of the commitment strategy according to

[4]This figure was originally created by Naga Krothapalli. Explicit permission was granted by the author to use this figure in this thesis.

environment factors is recommended in this meeting scheduling research. Despite the fact that both of these systems deal with multiple negotiation issues concurrently, the agents do not explicitly reason about the relationships among different issues under negotiation. For this reason, they can't propose offers or counter-offers to minimize the conflict and optimize the combined outcome. Nor do the agents explicitly reason about the relationships among different issues under negotiation, in order to propose offers or counter-offers (choose the appropriate parameters in the offer) to minimize the conflict and optimize the combined outcome. The ordering of different negotiation issues is not taken into consideration in either of these two approaches, which we feel is important for the agent to find a good negotiation approach.

In this thesis, we build a formalized model for the multi-linked negotiation problem, which is based on the analysis and characterization of the problem. Using this model, the agent can reason about the relationships among different negotiation issues and also evaluate different negotiation approaches. Additionally, a partial order scheduler is introduced so that an agent can reason about the influence of a commitment on one issue on other negotiating issues. It also can be used to reason about the flexibility of each commitment and how it affects the flexibility available for an agent to schedule (and reschedule) its local activities. Using this model and the partial order reasoning tools, the agent can search to find a good ordering of negotiation issues and their parameters and hence to improve its performance. Furthermore, we build a multi-leveled negotiation framework to facilitate multi-linked negotiation, which allows the agent to adjust its previously formed commitments to resolve conflicts.

1.4.2 Negotiation in a Complex Organizational Context

Another question addressed in this research is concerned with how to perform negotiation in a complex organizational context. Until now almost all related work can be categorized into two general classes: cooperative negotiation and competitive negotiation. In competitive negotiation, agents are self-interested and negotiate to maximize their own local utility; in cooperative negotiation, agents work to find a solution that increases their joint utility – the sum of the utilities of all involved agents. In the competitive negotiation class, significant work [51, 48, 50] has been done in the area of bounded rational self-interested agents (BRSI). Said agents are self-interested and social welfare is not a concern – each agent works to maximize its own utility though contracting, bidding and decommiting. In the cooperative negotiation class, significant work has been done in the area of conflict resolution through negotiation [15, 30, 59]. In these approaches there is no notion of individual agent utility – agents are "completely-cooperative" with each other and cooperate to solve problems together. Little work has been done to study negotiation between these two extreme cases.

We feel that as the sophistication of multi-agent systems increases, MAS will be neither simple market systems where each agent is purely self-interested, seeking to maximize its local utility, nor distributed problem solving systems where all agents are completely cooperative working to maximize their joint utility. This will occur for two reasons. First, agents from different separate organizational entities will come together to dynamically form virtual organizations/teams for solving specific problems that are relevant to each of

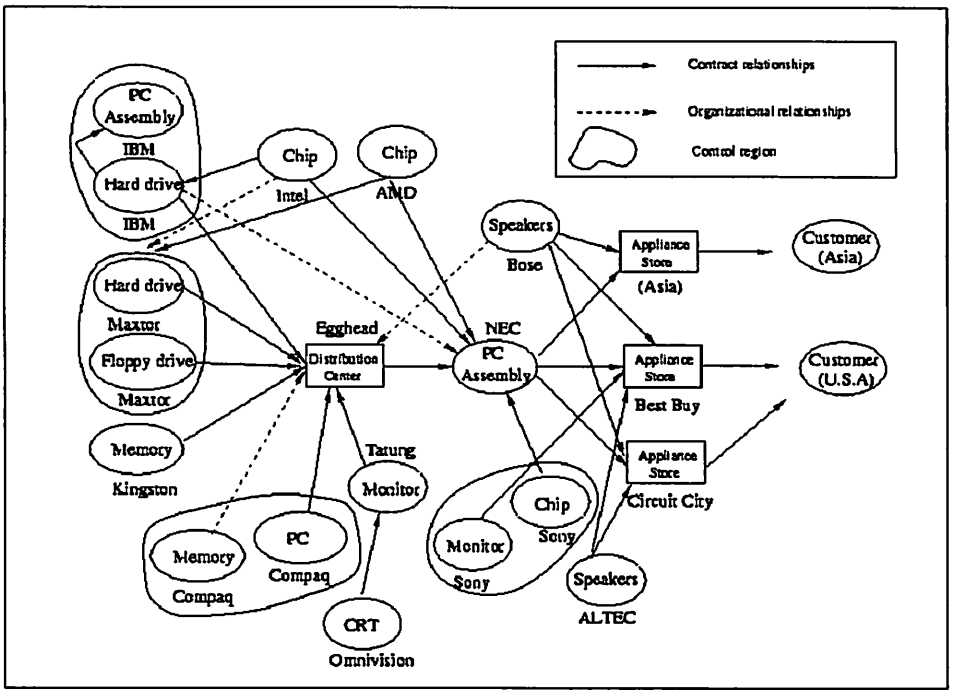


Figure 1.4: Supply chain example

their organizational entities [40]. How these agents work in their teams will often be dependent on the existence of both long term and short-term relationships and conform to their underlying organizational entities. Also, even for agents from self-interested organizations, it might be beneficial for them to be partially cooperative when they are in the situations where they will have repeated transactions with other agents from other organizational entities. Additionally, agents may be involved concurrently with more than one virtual organization while doing tasks for their own organizational entities.

Secondly, even agents working solely with agents of their own organizational entities will take varying attitudes in the spectrum of fully cooperative to totally self-interested in order for the organization to best achieve its overall goal. This perspective is based on a bounded-rational argument: it is not possible from a computational or communicational perspective for an agent to be fully cooperative, because the agent needs to take into account the utilities of all agents in the organization and the state of achievement of all organizational goals to be fully cooperative. Thus, it may be best for the organization to have agents being partially cooperative in their local negotiation with other agents rather than being fully cooperative in order to deal more effectively with the uncertainty of not having a more informed view of the state of the entire agent organization.

Given the complex organizational context in multi-agent systems, it is not enough for an agent simply to be purely self-interested or completely cooperative. It needs to have more flexible negotiation strategies between these two extreme cases, i.e. half-self-interested, half-cooperative; mostly self-interested, slightly cooperative; etc.

For example, let's consider the supply chain example in Figure 1.4. There are different organizational relationships among agents. For instance, there is an agent (agent_IBM_2) who produces hard drives, belonging to the IBM company. It provides hard drives for three different agents, with the following organizational relationships to it:

1. Agent_IBM_2 provides hard drives for the other agent (agent_IBM_1) that also belongs to IBM but assembles PC.

2. Agent_IBM_2 provides hard drives to an NEC agent (agent_NEC), and as the transactions between them become more frequent and regular, they form a virtual organization based on the recent transactions.

3. Agent_IBM_2 occasionally also provides hard drives for a distributor center (agent_DIS) based on a simple marketing mechanism.

When agent_IBM_2 negotiates with these three agents, it should use different negotiation strategies. When it negotiates with agent_IBM_1, it may need to be more cooperative than it is towards the other two agents if its most important goal is to increase the utility of IBM. However, even for the good of IBM's benefit, it may not be the best choice for agent_IBM_2 always to be accommodative towards agent_IBM_1. Sometimes it may bring IBM more profit for agent_IBM_1 to provide hard drives to agent_DIS rather than to agent_IBM_1. So the question is: how cooperative should agent_IBM_2 be towards agent_IBM_1?

When agent_IBM_2 negotiates with agent_NEC_1, it may need to be more cooperative than it is towards agent_DIS given the virtual organization it has formed with agent_NEC_1. How cooperative it should be depends on how important the utility increase of this virtual organization is to agent_IBM_2 and how the goal to increase the utility of this virtual organization relates to its other goals. Also, as we noticed before, the formation of this virtual organization is dynamic; it may also disappear sometime later if the environment changes, so agent_IBM_2 should adopt its negotiation strategy dynamically too.

When agent_IBM_2 negotiates with agent_DIS, should it be purely self-interested, given that there is only a simple marketing relationship between them? It may not be the case. There are two reasons. First, when there are multiple attributes involved in negotiation, it is possible to reach a win-win agreement rather than play a zero-sum game. Secondly, it is not good for an agent to try to maximize its own utility in every negotiation session from a long-term perspective; in other words, it may lose its long-term profit by being too aggressive every time (do you want to go back to the same sales person who tries to squeeze every penny out of your pocket?).

From the above examples we find it is necessary to have the following framework to support an agent's negotiation in a complex organizational context:

1. The agent can choose many different negotiation strategies in the spectrum from purely self-interested to accommodative. It should be easy for the agent to switch from one strategy to another strategy. A uniformed negotiation mechanism for all different negotiation strategies is an ideal choice.

2. The choice of negotiation strategy should not be hard-coded in the agent. The choice should depend on the agent's organizational goals, the current environmental circumstance, which agent is negotiated with, and what issue is under negotiation.

3. There should be no requirement of a centralized controller which coordinates the agent's behavior. The agent should be free to choose any negotiation strategy according to its goals set by its designer. Because every agent belongs to different users, it is not realistic to assume a centralized controller.

So far, there has been no such negotiation framework which provides the above capabilities for agents. Glass and Grosz [21] developed a measure of social consciousness called "brownie points" (BP). The agent earns BP each time it chooses not to default on a group task and loses BP when it does default for a better outside offer. The default of a group task may cause the agent to receive group tasks of less value in the future, hence reducing its long-term utility. The agent counts BP as part of it overall utility besides the monetary utility. A parameter $BPweight$ can be adjusted to create agents with varying levels of social consciousness. This work assumes there is a central mechanism controlling the assignment of group tasks according to the agent's rank (the agent's previous default behavior), which is not always appropriate for an open agent environment. Additionally, the "brownie points" only describe the agent's concern for the benefit of a pre-defined agent group. It is not flexible enough for the agent to choose a negotiation strategy for a specific agent concerning a specific issue.

In this thesis, we introduce an integrative negotiation mechanism which enables agents to manage all kinds of negotiation strategies in the spectrum from self-interested to completely cooperative in a uniform reasoning framework called the MQ framework. The agent not only can choose to be self-interested or cooperative, but also can choose how cooperative it wants to be. This provides the agent with the capability to adjust dynamically its negotiation strategy in a complex agent society. This mechanism allows the agent to quantitatively reason about each negotiation session so that it can choose an appropriate negotiation strategy. It bridges the gap between self-interested negotiation and cooperative negotiation.

Introducing this mechanism in the agent framework also strengthens the capability of multi-agent systems to model human societies. As we mentioned in Section 1.1, multi-agent systems are important tools for developing and analyzing models and theories of interactivity in human societies. There are many complicated organizational relationships in human society, and every person plays a number of different roles and is involved in different organizations. A multi-agent system with this integrative negotiation mechanism is an ideal test-bed to model human society and to study negotiation and organization theories.

1.4.3 Negotiation As a Distributed Search Process

The third topic we will consider is cooperative multi-step negotiation. In cooperative negotiation, agents negotiate to find a solution which increases their combined utility. Cooperative negotiation can be viewed as a distributed search process; both agents involved in the negotiation are searching for a solution which increases their combined utility. There

may be many solutions which increase the combined utility, some of which are better than others because they lead to a greater utility increase. We introduce a multi-step negotiation mechanism in which agents engage in a series of proposals and counter-proposals to find a solution which increases or maximizes the combined utility. A related question then arises: When should the negotiation process be stopped? More search effort may find a better quality solution; however, the negotiation process itself carries cost too. How can the agent balance the negotiation gain and cost? How much effort should the agent put into the negotiation process? The experimental result shows a phase transition phenomenon as the complexity of the negotiation situation changes. A measure of negotiation complexity is developed that can be used by an agent to choose the appropriate protocol, allowing the agents to explicitly balance the gain from the negotiation and the resource usage of the negotiation.

Furthermore, the agents negotiate over multiple attributes (dimensions) rather than over a single dimension. For example, agent A wants agent B to do task T for it by time 10, and requests the minimum quality of 8 for the task to be achieved. Agent B replies that it can do task T by time 10 but only with the quality of 6; however, if agent A can wait until time 15, it can get a quality of 12. Agent A will select the alternative it believes is better for both agents. The negotiation relates to both the completion time and achieved quality of the task, and thus the scope of the search space for the negotiation is increased, improving the agents' chance of finding a solution that increases the combined utility. A question raised here is: How should the multi-dimensional search be structured? How does the agent control the multi-dimensional negotiation process? How does the agent propose a new proposal given its previous proposal and the other agent's counter-proposal? We present two different search algorithms for this multi-dimensional negotiation process.

1.4.4 Summarizing Approaches

To attack the above problems, we need to address the research in all three categories of negotiation.

1. In the *decision* category, we use the motivational quantities (MQ) [67] framework to evaluate negotiation decisions. This framework provides the agent with the capability to reason about the relative importance of different goals and the values of achieving these goals based on organizational objectives. We use this framework to support an integrated negotiation model that allows agents to take different positions anywhere along the spectrum from the purely self-interested position to the completely cooperative position. As part of our use of this framework, we extend it through the use of a partial order scheduler. A partial order scheduler is exploited as a reasoning tool to allow the agent to handle concurrent, multi-linked negotiation issues and evaluate the flexibility and the feasibility in the negotiation.

2. In the *process* category, negotiation is performed at different abstraction levels. Rough commitments are formed at the upper level and then refined at the lower level to solve potential conflicts among different negotiation issues. Furthermore, a multi-step negotiation mechanism is built which allows the agents to find a better negotiation solution through additional negotiation effort.

3. In the *language* category, we develop techniques to allow negotiation over multiple issues such as the temporal scope of the commitment, the cost of the commitment, and the flexibility of the commitment.

After careful investigation of the above problems, it is recognized that each of them can be studied separately and more thoroughly. However, we have chosen to study all of them in this thesis with a wide angle view. The reason is that these problems are inter-related. For an agent which negotiates over a specific issue, it needs to consider how this negotiation is related to other issues under negotiation; this is the multi-linked negotiation problem. It also needs to choose appropriate negotiation attitude toward the other agent according to their organizational relationship; this is the integrative negotiation problem. Additionally, it needs to consider what protocol should be used for negotiation and how to balance the gain and the cost of the negotiation. Thus, as part of this thesis we have designed an agent architecture that handles these problems in an integrated way. This architecture supports not only sophisticated negotiation but also local agent control, planning, scheduling and execution.

The research in this thesis provides a road map to many of the issues involved in sophisticated negotiation. We provide solutions for certain of these problems, using a combination of heuristic and formal approaches. As a result, we hope we have transmitted to the readers the importance and the difficulties embedded with sophisticated negotiation, provided approaches for solving many of issues involved in such negotiation, and laid out other issues that need to be studied.

1.5 Results of this Research

The major contributions of this thesis work can be summarized as the following:

1. Creation of a formal model of the multi-linked negotiation problem. Using this model, an agent can reason about and analyze the interrelationships among negotiation issues and then choose the appropriate negotiation ordering and feature assignment for multiple related negotiation issues, and hence increase its local utility.

2. Construction of a negotiation-oriented partial order scheduler and related reasoning algorithm. Using this reasoning tool, the agent can explicitly reason about the temporal interrelationships among multiple negotiation issues.

3. Development of an integrative negotiation mechanism which allows an agent to choose any negotiation attitude from purely self-interested to completely cooperative. By quantitatively reasoning about the relationship between agents, it is possible to study more general types of negotiation besides the purely self-interested negotiation or the completely cooperative negotiation in a single framework. This enhances the ability of the multi-agent system to model more realistic problems and relationships.

4. Implementation of a multi-level negotiation framework which allows negotiation to be performed on different abstraction levels. Negotiation on the upper level builds

rough commitments which are the foundation for the global activity planning; negotiation on the lower level refines the commitments considering detailed constraints.

5. Introduction of the dimension of flexibility into the negotiation process and quantitatively reasoning about this dimension in the negotiation process. The agent can select different negotiation strategies according to the desired flexibility of a specific negotiation issue. The agent also can decide whether to maintain more flexibility given the prediction of the possible uncertainty of its local activities.

6. Creation of a multi-step, multi-dimensional cooperative negotiation process. Agents can negotiate over multiple issues such as the temporal scope of the commitment, and the quality to be achieved for the task, rather than over one single issue. It is possible to find a commitment that is good for both agents given their different problem solving contexts and hence to maximize their joint utility.

7. Evaluation of these approaches. Our approach to validate these contributions will be an experimental approach. We will implement these algorithms and mechanisms, and show how they perform on sample problems, with comparison to other alternative ways.

1.6 Reading this Dissertation

This thesis is organized as follows:

1. Chapter 1 introduces the necessary background information, the motivation and the topics for this thesis, and a high level view of the contributions.

2. Chapter 2 presents a formalized model for the multi-linked negotiation problem. It also details the related partial order reasoning tools. Experimental work shows how this management technique improves the agents' performance.

3. Chapter 3 details the integrative negotiation framework; experimental work shows how the choice of negotiation strategies affects the agent's utility and the social welfare.

4. Chapter 4 describes the multi-level negotiation framework. It shows how the negotiation is performed at different abstraction levels, and how the agent can balance its needs for flexibility to handle uncertainty with the cost of maintaining this flexibility.

5. Chapter 5 describes the multi-dimensional, multi-step negotiation for task allocation in a cooperative system. It details the mechanism and presents experimental results that show comparison of the performance of different protocols. It also compares different negotiation approaches that have been presented in this chapter and previous chapters, and discusses their implications on the agent's performance.

6. Chapter 6 discusses the related work and compares this work with other approaches.

7. Chapter 7 summarizes what has been accomplished and the contributions, and lays out directions for future research.

CHAPTER 2

MULTI-LINKED NEGOTIATION

In this chapter, we will first define the multi-linked negotiation problem (Section 2.1). Then we will present a partial order schedule and a set of related algorithms (Section 2.2), which are necessary for the agent to reason about the time constraints and the flexibility of each negotiation issue. The partial order schedule and the related reasoning tools also make the parallel negotiations feasible by eliminating the potential conflicts. Experimental work in Section 2.3 shows it is important for agents to reason about flexibility in the multi-linked negotiation problem. A formalized model for the problem is presented in Section 2.4. Using this model, the agent can find the best ordering of the negotiation issues and their parameters, and hence increase its local utility achievement. Experimental work is presented (Section 2.5) which shows that this management technique for multi-linked negotiation leads to improved performance. Section 2.6 summarizes this chapter.

2.1 Multi-Linked Negotiation Problem

2.1.1 Scenario and Basic Assumptions

Multi-linked negotiation deals with multiple negotiation issues when these issues are interconnected. In a multi-task, resource sharing environment, an agent may need to deal with multiple related negotiation issues including: tasks contracted to other agents, tasks requested by other agents, external resource requirements for local activities and interrelationships among activities distributed among multiple agents. These issues are related to each other. The result of one issue has influence on the possible solutions of other issues.

We use the following supply chain example to explain a multi-linked negotiation situation. However, the negotiation process and the following approach are domain-independent and not restricted to this application. Consider the following example shown in Figure 2.1.

1. *Consumer Agent* generates three types of new tasks: *Purchase_Computer* task for *Computer Producer Agent*, *Purchase_Parts* task for *Hardware Producer Agent*, and *Deliver_Product* task for *Transporter Agent*.

2. *Computer Producer Agent* receives the *Purchase_Computer* task from *Consumer Agent*, decides if it should accept this task and, if it does, what the promised finish time of the task should be. Figure 2.1 shows the local plan for producing computers; it includes a non-local task *Get_Hardware* that requires negotiation with *Hardware Producer Agent*. It also includes a non-local task *Deliver_Computer* that requires negotiation with the *Transporter Agent*.

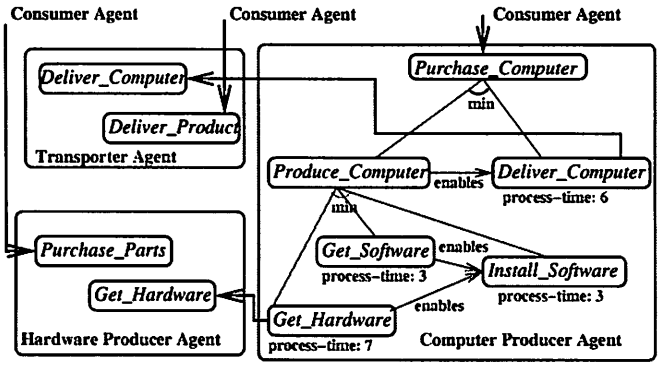


Figure 2.1: A supply chain scenario

3. *Hardware Producer Agent* receives two types of tasks: *Get_Hardware* from *Computer Producer Agent* and *Purchase_Parts* from *Consumer Agent*. It decides whether to accept a new task and what the promised finish time for the task is.

4. *Transporter Agent* receives receives two types of tasks: *Deliver_Computer* from *Computer Producer Agent* and *Deliver_Product* from *Consumer Agent*. It decides whether to accept a new task and what the promised finish time for the task is.

We first define two generalized terms to make the following description easier. In this thesis, the term *contractor agent* refers to the agent who performs the task for another agent and gets rewarded for successful completion of the task; *contractee agent* refers to the agent who has a task that needs to be performed by another agent and pays the reward to the other agent. The *contractor agent* and the *contractee agent* negotiate about the task and a contract is signed (a commitment is built and confirmed) if an agreement is reached during the negotiation.

In this work, the negotiation process between agents is based on the contract net model [60]:

1. *Contractee agent* announces a task by sending out a proposal.

2. *Contractor agent* receives this proposal, evaluates it, responds to it through one of three ways: accept it, simply reject it, or reject it but generate a counter-proposal.

3. *Contractee agent* evaluates the responses, it either chooses one accepted proposal to confirm, or chooses a counter-proposal to accept.

4. *Contractee agent* awards the task to the chosen *contractor agent* based on the commitment (the mutually accepted upon proposal or counter-proposal) which is confirmed by both agents; the negotiation process then ends successfully. If a mutually agreed proposal/counter-proposal can not be found, the negotiation process fails.

This process can be extended to a multiple-step process by introducing an extended series of alternative proposals and counter-proposals. However, in this chapter, we only

focus on the two-step (proposal, counter-proposal) negotiation process. We will discuss the implications of performing a multi-step negotiation instead of a two-step negotiation in Section 5.7.

The proposal to announce a task (t) that needs to be performed includes the following attributes:

1. *earliest start time* (est): the earliest start time of task t; task t can not be started before time est.

2. *deadline* (dl): the latest finish time of the task; the task needs to be finished before the deadline.

3. *minimum quality requirement* ($minq$): the task needs to be finished with a quality achievement no less than this requirement.

4. *regular reward* (r): if the task is finished as the contract requested, the contractor agent will get reward r.

5. *early finish reward rate* (e): if the contractor agent can finish the task by the time (ft) as it promised in the contract, it will get the extra early finish reward: $max(e * r * (dl - ft), r)$[1] in addition to the reward r.

6. *decommitment penalty rate* (p): if the contractor agent can not perform the task as it promised in the contract (i.e. the task could not finish by the promised finish time), it pays a *decommitment penalty* ($p * r$) to the contractee agent. If the contractee agent needs to cancel the contract after it has been confirmed, it needs to pay a *decommitment penalty* ($p * r$) to the contractor agent.

When the potential contractor agent receives a task proposal, it evaluates it and decides to either accept it or reject it. If it accepts this proposal, it needs to decide what the promised finish time should be. If it rejects the proposal, it can either simply say "no" or generate a counter-proposal which modifies some of the attributes in the proposal to accommodate its local problem solving context.

In the above discussion, we assume the negotiation is about a task which needs to be performed; however, the negotiation can also be about a resource requirement. The agent can require a resource during a time period and pay for this resource usage. In this situation, some of the attributes specified in the proposal are different from those in the above description[2], but the basic negotiation process is the same, and the methodologies we discuss in this chapter are also suitable for negotiation over resources.

[1]For each time unit the task finishes earlier than the deadline, the contractor agent gets extra reward $e * r$, but the total extra reward would not exceed the reward r.

[2]For example, the minimum quality requirement is not applicable for a resource requirement. A quantity requirement may be necessary to specify how much resource is needed.

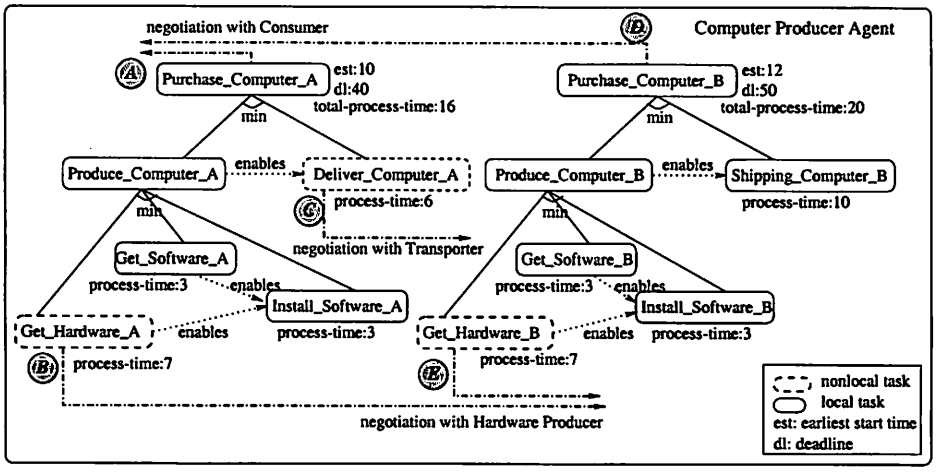


Figure 2.2: Computer producer agent's tasks

2.1.2 Introduction to the Problem

Suppose *Computer Producer Agent* has received the following two tasks:

task name : Purchase_Computer_A
arrival time: 5
earliest start time: 10 (arrival time + estimated negotiation time(5))[3]
deadline: 40
reward: r=10
decommitment penalty: p=0.5
early finish reward rate: e=0.01
task name : Purchase_Computer_B
arrival time: 7
earliest start time: 12 (arrival time + estimated negotiation time(5))
deadline: 50
reward: r=10
decommitment penalty rate: p=0.6
early finish reward rate: e=0.005

The agent's local scheduler (See Section 4.3.2) reasons about these two new tasks according to the above information: their earliest start times, deadline, estimated process times and the rewards. It then generates the following agenda which includes the accepted tasks:

Agenda 2.1.1 [10, 26] *Purchase_Computer_A* [26, 46] *Purchase_Computer_B*

This agenda is only a high level plan and does not include the execution details. The *Computer Producer Agent* checks the local plans for these tasks as shown in Figure 2.2 and finds there are five issues that need negotiation:

[3]The task should not start until the contract has been confirmed.

1. Negotiate with *Consumer Agent* about the promised finish time of *Purchase_Computer_A*[4].
2. Negotiate with *Consumer Agent* about the promised finish time of *Purchase_Computer_B*.
3. Negotiate with *Hardware Producer Agent* about *Get_Hardware_A*, concerning whether *Hardware Producer Agent* can accept this task and if the agent accepts this task, what the promised finish time is.
4. Negotiate with *Hardware Producer Agent* about *Get_Hardware_B*, with the same concerns as above.
5. Negotiate with *Transporter Agent* about *Deliver_Computer_A*, concerning whether *Transporter Agent* can accept this task, and if the agent accepts this task, what the earliest start time and the promised finish time for the task is.

These five issues are all related. The potential relationships among these negotiation issues can be classified as two types. One type of relationship is the *directly-linked* relationship: issue B affects issue A directly because issue B is a necessary resource (or a subtask) of issue A. The characteristics (such as cost, duration and quality) of issue B directly affect the characteristics of issue A. For example, as pictured in Figure 2.1[5], *Computer Producer Agent* receives task *Purchase_Computer_A* from *Consumer Agent*, and decides if it should accept this task and if it does, what the promised finish time of the task should be. The earlier the task is finished, the higher the reward *Computer Producer Agent* gets. In order to accomplish task *Produce_Computer_A*, *Computer Producer Agent* needs to generate an external request for hardware (*Get_Hardware_A*), and also needs to deliver the computer (*Deliver_Computer_A*) through a transporter agent. The negotiation on the task *Purchase_Computer_A* is directly linked to the negotiation on the two tasks: *Get_Hardware_A* and *Deliver_Computer_A*. If either one of these two tasks fails, the task *Purchase_Computer_A* can not be accomplished. Furthermore, when and how these two tasks are performed also affect the way the task *Purchase_Computer_A* is going to be accomplished. In the same way, the negotiation about *Get_Hardware_B* and *Purchase_Computer_B* are directly linked.

Another type of relationship is the *indirectly-linked* relationship: Issue A relates to issue B because they compete for use of a common resource. For example, as shown in Figure 2.2, besides the task *Purchase_Computer_A*, *Computer Producer Agent* has another contract on task *Purchase_Computer_B*. Because of the limited capability of the *Computer Producer Agent*, when task *Purchase_Computer_A* will be performed affects when task *Purchase_Computer_B* can be performed. The negotiation about task *Purchase_Computer_A* and the negotiation about task *Purchase_Computer_B* are *indirectly-linked.*

In general, *multi-linked negotiation* (including both the *directly-linked* and *indirectly-linked* relationships) describes situations where one agent needs to negotiate with multiple

[4]There are other attributes in the proposal that also can be negotiated over, such as *regular reward*, *earlier reward rate*, and *decommitment penalty*. We only mentioned *promised finish time* here as an example, because it is closely related to the negotiation on other issues.

[5]All task plans shown in this paper use the TÆMS language [16], which is also used in our implementation and experiments.

agents about different issues, where the negotiation over one issue influences the negotiations over other issues. The characteristics of the commitment on one issue affects the evaluation of a commitment or the construction of a proposal for another issue.

2.1.3 Analysis of the Problem

How can the agent deal with these multiple related negotiation issues? Two questions need to be answered here. The first question is in what order should the negotiation on each issue be performed? Should all the negotiations be performed concurrently or in sequence? If in sequence, in what sequence? There are potentially many other choices to order negotiations, such as doing some of them in parallel and some of them in sequence.

Why is the order of negotiation important? First, because each negotiation issue has a negotiation deadline, set by the contractee agent, if the contractor agent can not reply to a task proposal before the negotiation deadline, the negotiation fails. One reason for missing the negotiation deadline is that the contractor agent is busy on other negotiations before it decides to perform this negotiation. Furthermore, even if the negotiation is completed before its deadline, when the negotiation is started affects the outcome of the negotiation. For example, when there are several potential contractor agents, the earlier a response to negotiation is received, the more likely the offer is accepted. Likewise, the earlier the contractee agent initiates the negotiation, the more likely the contractor agent is to accept the proposal. Secondly, the earlier a negotiation is started, the larger the space for the agent to find a feasible solution is. For instance, given that the deadline for task *Get_Hardware_A* is 30, if the negotiation on this task finishes at time 10, there are 20 time units left for *Hardware Producer Agent* to arrange this task; if the negotiation finishes at time 20, *Hardware Producer Agent* only has 10 time units to execute this task. So the order of negotiation directly affects the outcome of the negotiation.

The second question is how the agent assigns values for those attributes (also referred as "features") in negotiation in terms to minimize the conflicts and maximize the utility. There are several features that the agent needs to negotiate over for each issue. For a task proposal, the contractee agent needs to find the earliest start time and deadline to request for the task, how much reward to pay for this task, the early reward rate, the decommitment penalty, etc. The contractor agent needs to decide the promised finish time. Some of these features are related to the features of other issues. For example, the deadline proposed for task *Get_Hardware_A* affects the earliest start time of task *Deliver_Computer_A*, and the deadline of task *Deliver_Computer_A* affects the promised finish time for task *Purchase_Computer_A*. The agent needs to find appropriate values for these features to avoid conflicts among them and to make sure there is a feasible local schedule to accommodate all the local tasks and commitments. Furthermore, the values of these features influence the outcomes of the negotiation and the agent's local utility. For example, the greater the reward is, the greater the likelihood that the task will be accepted by the contractor agent; however, the contractee agent's local utility decreases as the reward it pays to the contractor agent increases. Also, the later the deadline for task *Get_Hardware_A* is, the higher the possibility that the task will be accepted by the *Hardware Producer Agent*; however, it leaves less freedom for task *Deliver_Computer_A*, and it also makes the promised fin-

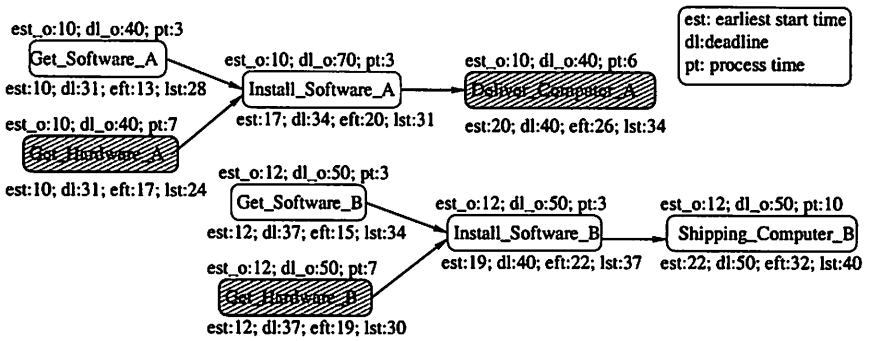


Figure 2.3: The partial order schedule of *Computer Producer Agent*

ish time for task *Purchase_Computer_A* later, hence reducing the early reward *Computer Producer Agent* may get.

A good negotiation approach for a multi-linked negotiation problem should provide the agent with an appropriate negotiation order of all issues and a feature assignment (assigns a value to every attribute that needs to be decided in this problem) for those attributes under negotiation. Using such a negotiation approach, the agent can avoid the conflicts among negotiation issues and optimize its utility achievement. In Section 2.2 we will introduce a partial order scheduler which allows the agent to reason about the time-related constraints and the flexibility on each negotiation issue. The experimental work in Section 2.3 shows that it is important for the agent to manage the flexibilities in multi-linked negotiation. A formalized model of the multi-linked negotiation problem is presented in Section 2.4; two search algorithms are developed which allow the agent to find a good negotiation approach. The experimental work in Section 2.5 shows that good negotiation approaches improve the agent's performance significantly.

2.2 Partial Order Schedule and Related Algorithms

2.2.1 Partial Order Schedule

A partial-order schedule is the basic reasoning tool that we use for multiple related negotiations. Here we present the formalization of the partial-order schedule and use examples to explain how it works for a multi-linked negotiation. Figure 2.3 shows the partial-ordered schedule from the example in Figure 2.2.

A *partial order schedule*[6] represents a group of tasks with specified precedence relationships among them using a directed acyclic graph: $G = (V, E)$. $V = \{u\}$, when each vertex in V represents a task. $E = \{(u, v) | u, v \in V)\}$. Each edge (u, v) in E denotes the precedence relationship between task u and task v ($P(u, v)$), that means task u has to be finished before task v can start.

[6]In this thesis, the term "partial order schedule" refers a representation of a group tasks with specified precedence relationships, which also includes the associated definitions in this Section. In this thesis, we use the term "partial order scheduler" to refer the procedure which actually produces partial order schedules for tasks, and a set of associated reasoning algorithms presented in Section 2.2.2.

Task is represented as a node in the graph; it is the basic element of the schedule. A task t needs a certain amount of process time, also referred as its duration ($t.process\_time$). A task can be a local task or a non-local task; a local task is performed locally (i.e, task *Get_Software_A* and task *Deliver_Computer_A*) and a non-local task (i.e. the *Get_Hardware _A* task) is performed by another agent; hence, it does not consume local process time.

The *pretasks of task* t is a set of tasks that need to be finished before t can start: $Pre(t) = \{s | s \in V \wedge (s,t) \in E\}$; t can start only after all tasks in $Pre(t)$ have been finished. For example, the pretasks of task *Install_Software_A* includes task *Get_Hardware _A* and task *Get_Software_A*.

The *posttasks of task* t is a set of tasks that only can start after t has been finished: $Post(t) = \{r | r \in V \wedge < t, r > \in E\}$. For example, the posttasks of task *Install_Software_B* includes task *Shipping _Computer _B*.

A task t has constraints of *earliest start time* ($t.est$) and *deadline* ($t.dl$). The *earliest start time* of a task t ($t.est$) is determined by the *earliest finish time* of its pretasks ($eft[Pre(t)]$) and its *outside earliest start time* constraint ($t.est\_o$) [7]:

$$t.est = max(eft[Pre(t)], t.est\_o)$$

The *earliest finish time* of a task t ($t.eft$) is defined as:

$$t.eft = t.est + t.process\_time$$

The *earliest finish time* of a set of tasks V ($eft[V]$) is defined as the earliest possible time to finish every task in the set V; it depends on the *earliest start time* and the duration of each task. For example, in Figure 2.3, the *outside earliest start time* constraint for task *Install_Software_A* is 10 (same as its super task *Purchase_Computer A*), the *earliest finish time* for its pretasks is 17 (assume *Get_Hardware_A* could be finished at its earliest possible time), then the *earliest start time* for task *Install_Software_A* is 17.

The *deadline* of task t ($t.dl$) is determined by the *latest start time* of its posttasks ($lst[Post(t)]$) and its *outside deadline* constraint ($t.dl\_o$):

$$t.dl = min(lst[Post(t)], t.dl\_o);$$

The *latest start time* of a task t ($lst(t)$) is defined as:

$$t.lst = t.dl - t.process\_time;$$

The *latest start time* of a set of tasks V ($lst[V]$) is defined as the latest time for the tasks in this set to start without any task missing its deadline. It depends on the deadline and the duration of each task.

The *flexibility of task t* represents the freedom to move the task around in this schedule. $F(t) = \frac{t.dl - t.est - t.process\_time}{t.process\_time}$.
For example, $F(Get\_Software\_A) = \frac{40-10-3}{3} = 9$.

[7] *outside earliest start time* for task t is the earliest possible start time decided by the problem solving context. As a given parameter, it is not changeable during the partial order reasoning process. For example, if the current time is 15, the task can not start before time 15. In a similar way, the *outside deadline* constraint is the task's *deadline* decided by the problem solving context.

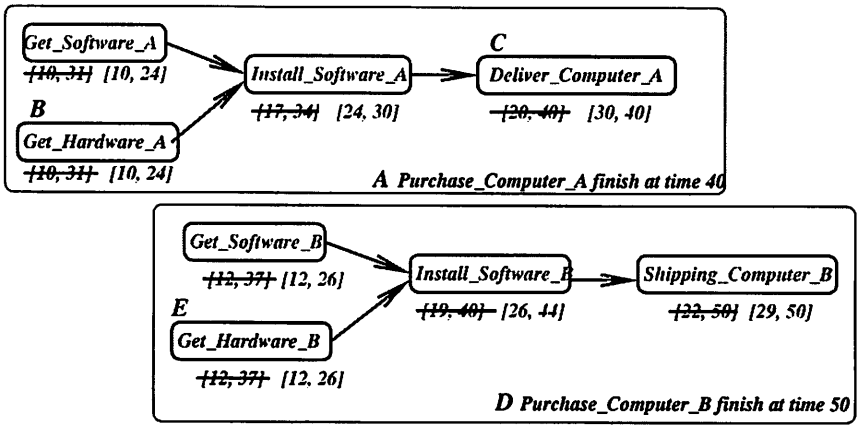


Figure 2.4: The consistent ranges for negotiation

The *flexibility of a schedule S* measures the overall freedom of this schedule. It is the sum of the flexibility of each activity weighted by its process time of the total process time of the schedule. The flexibility of the task with a longer process time has a larger influence on the flexibility of the schedule.
$F(S) = \sum_{t \in S} F(t) * \frac{t.process\_time}{(\sum_i t_i.process\_time)}$

A *feasible linear schedule* is a total ordered schedule of all tasks, that fulfills the following conditions:

- Each task t takes n (n>=1, if t is interruptible; otherwise, n=1.) time periods ($p_i, i = 1, ...n$) for execution, $\sum_i p_i = t.processtime$.
- All precedence relationships are valid.
- All earliest start time and deadline constraints are valid.

A partial-order schedule is a *valid partial order schedule* if there exists at least one *feasible linear schedule* that can be produced from this partial order schedule without additional constraints and with the interruptible execution assumption[8].

Without additional constraints and with the interruptible execution assumption, for a task t with the range [est, dl], no matter what time t is executed during this range, if there exists at least one feasible linear schedule that can be produced from this partial schedule, then the range [est, dl] for t is a *free range* because task t can be executed during any period in this range.

[8]Partial-Order Schedule is a representation and reasoning tool of a group of tasks and their interrelationships. It is not an executable schedule for the agent. To translate a partial-order schedule to an executable linear schedule, there are two different assumptions: the task is interruptible or un-interruptible. The interruptible execution assumption is that the agent can switch to another task during the execution of one task, and it can switch back at some point and continue the execution of the incomplete task. The un-interruptible execution assumption does not allow execution of a task to be split into parts. In our work here we adopt the interruptible execution assumption.

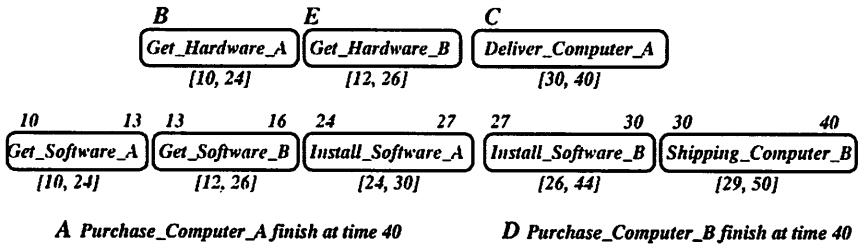


Figure 2.5: The feasible linear schedule

Without additional constraints and with the interruptible execution assumption, for a set of tasks t_i, $(i = 1, 2, ..., n)$, with the range $[est_i, dl_i]$, $(i = 1, 2, ...n)$ respectively, no matter what time t_i is executed during the range $[est_i, dl_i]$, if there exists at least one feasible linear schedule that can be produced from this partial schedule, then the ranges $[est_i, dl_i]$, $(i = 1, 2, ...n)$ for tasks t_i, $(i = 1, 2, ..., n)$ are *consistent ranges*. Negotiation over tasks t_i, $(i = 1, 2, ..., n)$ can be performed during these consistent ranges in parallel without worrying about conflicts. Figure 2.4 shows the consistent ranges for the tasks in the supply chain example. Figure 2.5 presents the *feasible linear schedule* given these consistent ranges.

The partial order schedule work is related to the Graphical Evaluation and Review Technique(GERT) [41] which is used for project scheduling and management. The big difference between the GERT work and ours is that the GERT work is not oriented to negotiation; all activities are local and can be managed with authority. Thus, with GERT there is no reasoning about free range, consistent ranges and schedule flexibilities which we feel are critical for an agent to effectively manage multi-linked negotiation.

2.2.2 Algorithms

We build the following algorithms to support the negotiation based on the partial order schedule. We only describe the functions of these algorithms, the detailed processes are presented in Appendix B.

Algorithm 2.2.1 *Propagate_EST_DL*
Given a set of tasks with the outside constraints of the earliest start times and deadlines, the durations and the precedence relationships, this procedure finds the earliest start time (t.est) and the deadline (t.dl) for each task t according to the definitions in Section 2.2.1.

Algorithm 2.2.2 *Get_Earliest_Finish_Time*
Given a set of tasks V, each task t has earliest start time (t.est) and its duration (t.process_time), this procedure calculates the earliest finish time of a set of tasks V ($eft[V]$).

Algorithm 2.2.3 *Get_Latest_Start_Time*
Given a set of tasks V, each task t has its deadline (t.dl) and its duration (t.process_time), this procedure calculates the latest start time of a set tasks V ($lst[V]$).

Algorithm 2.2.4 *Feasible_Schedule*
Given a partial order schedule (V, E), each task has its earliest start time and duration with respect to its pretask, posttask and its outside constraints, this procedure generates a feasible linear schedule if the partial order schedule is valid; otherwise it reports failure.

Theorem 2.2.1 *If there exists a feasible linear schedule, the Feasible_Schedule algorithm can find one.*

The proof of this theorem is presented in Appendix C.

Besides Algorithm 2.2.4, we also develop Algorithm 2.2.5 to answer the question of whether a partial order schedule is valid without trying to find a feasible linear schedule.

Algorithm 2.2.5 *Range_Evaluation*
This procedure determines if a partial order schedule is valid.

The basic idea of Algorithm 2.2.5 is to check every possible time range $[est, dl]$ by constructing all possible combinations of every task's earliest start time and deadline. For all tasks falling into this range, if the sum of process times of these tasks is greater than the time available $(dl - est)$, there is no feasible linear schedule; otherwise, there exists a feasible linear schedule, because every task t can find a place between its earliest start time and its *deadline*.

This proves the following theorem:

Theorem 2.2.2 *A partial order schedule is valid if and only if the procedure 2.2.5 returns true.*

Using the above procedure, we have the following algorithm to find the free range of a non-local task used for the negotiation.

Algorithm 2.2.6 *Find_NL_Range*
Given a partial order schedule $G(V, E)$ containing a task nlt, this procedure finds the biggest free range for task nlt.

If there is more than one non-local task, we need to sort them according to some characteristics (i.e. flexibility, importance, difficulty of negotiation, etc.), and work on them one by one. When the Find_NL_range procedure works on one task nlt_i, the range for those tasks before it (nlt_1, ..., $nlt\_(i-1)$) has already been decided and can not be changed. The range for those tasks after it ($nlt\_(i+1)$, ...) are set to a range that is as small as possible, so as to allow this task nlt_i to have more freedom.

All of the above algorithms and procedures provide a toolkit for the agent to reason about its proposals and evaluate counter-proposals from other agents.

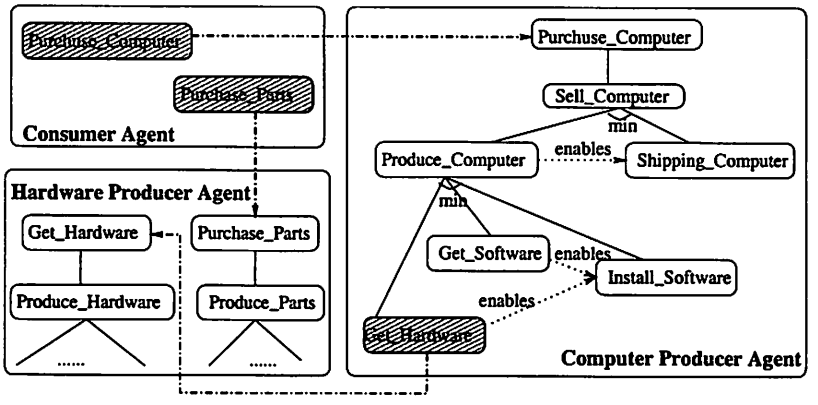


Figure 2.6: Three agents scenario

2.3 Experiments on Flexibilities

We have implemented an agent architecture including the agent controller, agent negotiator and execution components. All the above algorithms and procedures associated with reasoning about the partial-order schedule have been implemented so as to enable the reasoning in the multi-linked negotiation process as indicated in the examples described in this chapter. We designed the following experiments to study how the different flexibility policies in negotiation, which involve different types of reasoning strategies, affect the agent's performance.

The experimental environment is set up based on the scenario described in Figure 2.6. It is a simplified scenario from the example shown in Section 2.1.1. Three agents were built using the JAF agent framework [62]. New tasks were randomly generated with decommitment penalty rate $p \in [0, 1]$, early finish reward rate $e \in [0, 0.1]$, and deadline $dl \in [45, 105]$, and arrived at the contractor agents periodically. The local scheduler of the agent schedules all incoming new tasks according to their earliest start times, deadline, process times and rewards and generates an agenda (such as agenda 2.1.1) including the accepted tasks. From this agenda, the agent can find the scheduled finish time of each task. It could continue the negotiation about these incoming tasks just based on the information from this agenda without further reasoning about the detailed plan for each task (actually, that is what the agent does when using the "Earliest-Finish-Time Policy" and the "Deadline Policy"). At the same time, if the local plan of these accepted tasks involves any non-local task *nlt*, then the Find_NL_Range procedure (Algorithm 2.2.6) is used to find the earliest start time and the deadline of the task nlt. The agent would then start negotiation with the other agent about task *nlt* based on this time range.

In this experiment, *Computer Producer Agent* needs to deal with the multi-linked negotiation issues related to the incoming task *Purchase_Computer* and the outgoing task *Get_Hardware*. The following three different negotiation policies were tested:

1. *Earliest Finish Time Policy*: the agent finds the scheduled finish time of the task from its agenda and promises it as the finish time in the contract with the intention of maximizing the early finish reward.

Table 2.1: Comparison of performance using different negotiation policies in multi-linked negotiation CPA: *Computer Producer Agent*; **HPA:** *Hardware Producer Agent*

	Policy	Tasks Received	Tasks Accepted	Tasks Canceled	Early Finished	Decommit Penalty	Early Reward	Utility
CPA	1	60	59	27	33	123	283	391
CPA	2	60	60	0.5	0	2.9	0	413
CPA	3	60	60	1.7	53	8.3	297	697
HPA	1	87	87	27	29	0	36	268
HPA	2	84	84	9.6	0	0	0	256
HPA	3	87	87	11	17	0	32	294

2. *Deadline Policy*: The agent promises the finish time which is the same as the *deadline* of the task with no consideration of the early finish reward.

3. *Flexibility Policy*: the agent analyzes its detailed partial-order schedule. If non-local tasks are found, it arranges reasonable flexibility (1, in this experiment) for each non-local task, and based on this arrangement, the finish time of the incoming task is decided and promised to the contractee agent.

In all three cases above, the multiple negotiations are performed concurrently based on the free ranges found by the partial-order scheduler. However, with the first two policies, the agent does not reason about the interaction among issues or manage the flexibilities for each issue.

The experiments are performed in the MASS simulator environment [63]. Every group experiment runs 3 times for 1000 time clicks each, each time using one of the three different polices. The entire experiment contains 32 group experiments. We use the same scenario and task structures as described in Figure 2.6, tasks vary with randomly generated parameters. This scenario represents a class of problem where one agent needs to deal with both directly and indirectly related negotiation problems. The deadlines of tasks are randomly generated from a range which allow there are possible flexibilities available for those sub-contracted tasks.

Table 2.1 shows the comparison of the agents' performance using different policies. For *Computer Producer Agent*, who has multi-linked negotiation issues, the flexibility policy is obviously better than the other two policies; it gets more early reward and pays fewer decommitment penalties.[9] For *Hardware Producer Agent*, the *Earliest Finish Time Policy* and the *Flexibility Policy* make no difference to the agent's decision making processes, since the agent has no sub-contracted task that needs consideration. The reason that the *Earliest*

[9]Using a t-test, with the 0.01 alpha-level, the following hypothesis H_o is rejected: when using the flexibility policy, *Computer Producer Agent* achieves an extra utility that is equal to 64% of the utility gained when using the *Earliest Finish Time Policy*, compared to the hypothesis H_a: when using the flexibility policy, *Computer Producer Agent* achieves an extra utility that is more than 64% of the utility gained when using the *Earliest Finish Time Policy* .

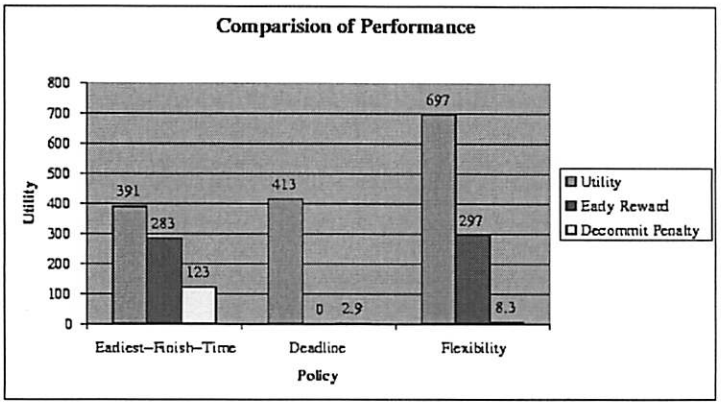


Figure 2.7: Comparison of performance using different negotiation policies in graph

Finish Time Policy provides less utility is because the *Computer Producer Agent* cancels more task requests (because the finish times *Hardware Producer Agent* could provide are too late), and hence *Hardware Producer Agent* has fewer tasks to perform and gains less reward. These experiments shows that in a multi-linked negotiation situation, it is very important for the agent to reason about the relationships among different negotiation issues and maintain reasonable flexibility for them. This type of reasoning decreases the likelihood of decommitment for previously settled issues and thus gains more utility. Figure 2.7 shows the utility, early reward, and decommitment penalty of *Computer Producer Agent* using three different policies.

2.4 Model of the Problem

In the previous section, we presented the partial order schedule and the related algorithms, which enable the agent to reason about the time-related constraints on each negotiation issue and manage the flexibility. However, this toolkit does not answer the following questions: in what order should the negotiations be performed and generally, what is a good feature assignment for all the attributes in negotiation. In this section, we first introduce a formalized model of the multi-linked negotiation problem and then present two search algorithms to find a good negotiation approach: a feature assignment and an order for a group of negotiations that an agent needs to conduct in order to complete its tasks.

2.4.1 Definition of the Problem

A multi-linked negotiation problem occurs when an agent has multiple negotiation issues that are interrelated.

Definition 2.4.1 *A* **multi-linked negotiation problem** *is defined as an undirected graph (more specifically, a forest as a set of rooted trees):* $\mathcal{M} = (V, E)$*, where* $V = \{v\}$ *is a finite set of negotiation issues, and* $E = \{(u, v)\}$ *is a set of binary relations on* V*.* $(u, v) \in E$ *denotes that the negotiation on u and the negotiation on v are directly-linked. Each negotiation issue* $v_i (v_i \in V)$ *is associated with a set of attributes* $\mathcal{A}_i = \{a_{ij}\}$*. Each attribute* a_{ij} *either already has been determined (already has a value) or needs to be decided (to be assigned to a value). The relationships among the negotiation issues are described by a forest, a set of rooted trees* $\{T_i\}$*. There is a relation operator associated with every non-leaf issue v on the tree (denoted as* $\rho(v)$*), which describes the relationship between this issue v and its children. This relation operator has two possible values:* AND *and* OR*.*

A negotiation issue v may be a task to be allocated or a resource to be acquired through negotiation. Multiple resources requested for a single task can be modeled as the following. The task is represented as a node t with multiple children nodes, each represented as a separate negotiation on one resource; the relation operator associated with t has the value AND.

From an agent's viewpoint, there are two types of negotiation issues:

1. **Incoming negotiation issue**: A task proposed by another agent. For example, issue A (*Purchase_Computer_A*) and D (*Purchase_Computer_B*) in Figure 2.2 are incoming negotiation issues for *Computer Producer Agent*.

2. **Outgoing negotiation issue**: A task needed to be sub-contracted to another agent, or a resource requested for a local task. For example, issue B (*Get_Hardware_A*), C (*Deliver_Computer_A*) and E (*Get_Hardware_B*) in Figure 2.2 are outgoing negotiation issues for *Computer Producer Agent*.

Definition 2.4.2 *A negotiation issue v is* **successful** *(denoted as* $S(v)$*) if and only if a commitment has been established and confirmed for this issue by those agents which are involved in this negotiation.*

Definition 2.4.3 *A leaf node v is* **task-level successful** *(denoted as* $TS(v)$*) if and only if v is* **successful** *(*$S(v) = true$*); A non-leaf node v is* **task-level successful** *(denoted as* $TS(v)$*) if and only if the following conditions are fulfilled:*

- *v is* **successful** *(*$S(v) = true$*);*
- *all its children are* **task-level successful** *if* $\rho(v) = AND$*; or at least one of its children is* **task-level successful**, *if* $\rho(v) = OR$*.*

For each negotiation issue v_i, $p_s(v_i)$ denotes the **success probability**, the probability that v_i is **successful** ($p(S(v_i) = true)$). There is a function ζ_i mapping the values of the attribute a_{ij}, $j = 1, 2, ..., k$, to $p_s(v_i)$: $p_s(v_i) = \zeta_i(a_{i1}, a_{i2}, ..., a_{ik})$.

$\beta(v_i)$ denotes the **decommitment penalty** [48] of v_i. If v_i is **successful** ($S(v_i) = true$), but v_k isn't **task-level successful** ($TS(v_k) = false$), where v_k is the root of the tree that v_i belongs to (denoted as $v_k = root(v_i)$), the utility of the agent decreases by the amount of $\beta(v_i)$ (it represents the penalty paid to the other agent which is involved in the negotiation of issue v_i.) There is a function η_i mapping the values of the attribute a_{ij}, $j = 1, 2, ..., m$, to $\beta(v_i)$: $\beta(v_i) = \eta_i(a_{i1}, a_{i2}, ..., a_{im})$.

If v_i is a root of a tree, then $\gamma(v_i)$ denotes the **task-level successful reward** of v_i. The agent's utility increases by the amount of $\gamma(v_i)$ when v_i is **task-level successful**. There is a function θ_i mapping the values of the attribute a_{ij}, $j = 1, 2, ..., m$, to $\gamma(v_i)$: $\gamma(v_i) = \theta_i(a_{i1}, a_{i2}, ..., a_{im})$.

The attributes of a negotiation issue are domain dependent. They are specified according to the application domain. The above functions ζ_i, η_i and θ_i are also defined according to the application domain. However, there are some common attributes introduced here:

1. negotiation duration ($\delta(v_i)$): the time needed for the negotiation on v_i to get a result, either success or failure.

2. negotiation start time ($\alpha(v_i)$): the start time of the negotiation on v_i. $\alpha(v_i)$ is an attribute that needs to be decided by the agent.

3. negotiation deadline ($\epsilon(v_i)$): the negotiation on v_j needs to be finished before this deadline $\epsilon(v_i)$. The negotiation is no longer valid after time $\epsilon(v_i)$, which is the same as a failure outcome of this negotiation. For example, if a task v_i is proposed for negotiation, the contractee agent needs to reply before time $\epsilon(v_i)$. Otherwise, this task proposal is no longer valid and the contractee agent would think the contractor agent is not interested in this task. Furthermore, even if the agent starts the negotiation before $\epsilon(v_i)$, it is not necessarily true that all times before $\epsilon(v_i)$ are equally good. Usually, an earlier started negotiation has a better chance to succeed for two reasons: the other party considers this issue before other later arriving issues, and this issue has a bigger time range for negotiation. This relationship is described by the function ζ_i that takes $\alpha(v_i)$ as one of its parameters.

2.4.2 Description of the Solution

Given this multi-linked negotiation problem $\mathcal{M} = (V, E)$, an agent needs to make a decision about how the negotiation over these issues should be performed. The decision concerns the following two questions:

1. How should the negotiation over each issue be ordered? Should the negotiation over each issue be parallel or sequenced? If sequenced, in what order? Or should some of them be performed in parallel, while others be sequenced?

2. What values should be assigned to the attributes of each issue that need to be decided in negotiation?

Definition 2.4.4 *A* **negotiation ordering** ϕ *is a directed acyclic graph (DAG),* $\phi = (V, E_\phi)$. *If* $e : (v_i, v_j) \in E_\phi$, *then the negotiation on* v_j *can only start after the negotiation on* v_i *has been completed.* $e : (v_i, v_j)$ *is being referred to as a* **partial order relationship (POR),** e. *A* **negotiation ordering** *can be represented as a set of* **PORs,** $\{e\}$.

Definition 2.4.5 *A negotiation schedule* $\mathcal{NS}(\phi)$ *contains a set of negotiation issues* $\{v_i\}$. *Each issue* v_i *has its negotiation start time* $\alpha(v_i)_\phi$ *and its negotiation finish time* $\varepsilon(v_i)_\phi$ *that is calculated based on its negotiation duration* $\delta(v_i)$ *and its negotiation start time* $\alpha(v_i)_\phi$.

Using the topological sorting algorithm, a negotiation schedule $\mathcal{NS}(\phi)$ can be generated from a negotiation ordering ϕ assuming all negotiation issues started at their earliest possible times[10]. Given this assumption and a start time τ[11] for a set of negotiation issues, the negotiation schedule generated from a negotiation ordering is unique.

Definition 2.4.6 *Given a start time* τ, *a negotiation ordering* ϕ *is* **valid** *if for every negotiation issue* v_i, *the finish time* $\varepsilon(v_i)_\phi$ *is no later than the negotiation deadline* $\epsilon(v_i)$.

The number of possible negotiation orderings for n negotiation issues $S(n)$ is: $S(n) = G(n) - I(n)$

$$G(n) = 3^{\binom{n}{2}}.$$

$G(n)$ denotes the number of all possible different directed graphs based on n vertices. There are three possibilities: there is no edge between them, $(v_i, v_j) \in E$, or $(v_j, v_i) \in E$. $I(n)$ denotes the number of graphs that contain cycles. $S(n) = O(3^{\frac{n^2}{2}})$

Definition 2.4.7 *A feature assignment* φ *is a mapping function that assigns a value* μ_{ij} *to each attribute* a_{ij} *that needs to be decided in the negotiation. For those attributes that already have been decided, value* μ_{ij} *is the decided value.*

Definition 2.4.8 *A feature assignment* φ *is* **valid** *if given the assigned values of those attributes, there exists at least one feasible local schedule for all tasks and negotiation issues. It is assumed there is a function that can test if a feature assignment* φ *is valid; this function is domain dependent.*

Definition 2.4.9 *A negotiation approach* (ϕ, φ) *is a combination of a valid negotiation ordering* ϕ *and a valid feature assignment* φ.

The evaluation of a negotiation approach is based on the expected task-level successful rewards and decommitment penalties given all possible negotiation outcomes for each negotiation issue.
A negotiation issue has two possible outcomes: **success** and **failure.**

[10]It assumes the negotiation on an issue starts right after all the negotiations on those issues that precede this issue have been finished according to the negotiation ordering.

[11]The start time specifies the earliest start time for the negotiations on this set of issues. It is also possible to specify the earliest start time for each negotiation separately.

Definition 2.4.10 *A negotiation outcome χ for a set of negotiation issues $\{v_j\}, (j = 1, ..., n)$ is a set of numbers $\{o_j\}(j = 1, ..., n), o_j \in \{0, 1\}$. $o_j = 1$ means v_j is successful, $o_j = 0$ means v_j fails. There are a total of 2^n different outcomes for n negotiation issues, denoted as $\chi_1, \chi_2, \cdots \chi_{2^n}$.*

Definition 2.4.11 *The expected value of a negotiation approach (ϕ, φ), denoted as $\mathcal{EV}(\phi, \varphi)$, is defined as: $\mathcal{EV}(\phi, \varphi) = \sum_{i=1}^{2^n} P(\chi_i, \varphi) * (R(\chi_i, \varphi) + C(\chi_i, \phi, \varphi))$*

$P(\chi_i, \varphi)$ denotes the probability of the outcome χ_i given the feature assignment φ.
$P(\chi_i, \varphi) = \prod_{j=1}^{n} p_{ij}(\varphi)$

$$p_{ij}(\varphi) = \begin{cases} p_s(v_j), (p_s(v_j) = \zeta_j(\varphi)) & \text{if } o_j \in \chi_i = 1 \\ 1 - p_s(v_j) & \text{if } o_j \in \chi_i = 0 \end{cases}$$

$R(\chi_i, \varphi)$ denotes the agent's utility increase given the outcome χ_i and the feature assignment φ. $R(\chi_i, \varphi) = \sum_j \gamma(v_j) = \sum_j \theta_j(\varphi)$
v_j is a root of a tree and v_j is **task-level successful** *($TS(v_j) = true$) according to the outcome χ_i.*

$C(\chi_i, \phi, \varphi))$ denotes the decommitment penalty according to the outcome χ_i, the negotiation orderingϕ and the feature assignment φ.
$C(\chi_i, \phi, \varphi)) = \sum_j \beta(v_j) = \sum_j \eta_j(\varphi)$
v_j represents every negotiation issue that fulfills all the following conditions:

1. *v_j is successful according to χ_i;*
2. *the root of the tree that v_j belongs to isn't* **task-level successful** *according to χ_i;*
3. *according to the negotiation ordering ϕ, there is no such issue v_k existing that fulfills all the following conditions:*
   (a) *v_k and v_j belong to the same tree;*
   (b) *v_k gets a failure outcome according to the outcome χ_i;*
   (c) *v_k makes it impossible for $root(v_j)$ to be* **task-level successful;**
   (d) *the negotiation finish time of v_k ($\varepsilon(v_k)_\phi$) is no later than the negotiation start time of v_j ($\alpha(v_j)_\phi$) according to the negotiation ordering ϕ.*

2.4.3 Description of the Algorithm

Based on the above definition, we present two search algorithms that find a good negotiation approach for a multi-linked negotiation problem $\mathcal{M} = (V, E)$.

2.4.3.1 Complete Search

Algorithm 2.4.1 *Find the best negotiation approach.*
Input: $\mathcal{M} = (V, E)$, the start time for negotiation τ, a set of valid feature assignments $\omega = \{\varphi_k\}$, $k = 1, ..., m$.
Output:the best negotiation approach.

Generate all valid negotiation orderings $\{\phi_i\}$;
best_value = minimum_value;
best_ordering = null;
best_assignment = null;
for each negotiation ordering ϕ_i
    for each valid feature assignment φ_k
        if $\mathcal{EV}(\phi_i, \varphi_k) > best\_value$
            $best\_value = \mathcal{EV}(\phi_i, \varphi_k)$;
            $best\_ordering = \phi_i$;
            $best\_assignment = \varphi_k$;
return (best_ordering, best_assignment)

If the set of valid feature assignments is a complete set of all possible valid feature assignments, Algorithm 2.4.1 is guaranteed to find the best negotiation approach. However, when the attributes have continuous value ranges, it is impossible to find all possible valid feature assignments. The following depth-first search (DFS) algorithm searches over the entire value space for all undecided attributes by pre-defined search step size and finds a set of valid feature assignments.

Algorithm 2.4.2 *Find a set of valid feature assignments.*
Input: $\mathcal{M} = (V, E)$.
For each attribute a_{ij}, *if* a_{ij} *is already decided, the value of* a_{ij} *is* $defined\_value(a_{ij})$;
if a_{ij} *is undecided, the maximum possible range for* a_{ij} *is:* $[min\_value(a_{ij}), max\_value(a_{ij})]$,
the search step size: $step_{ij}$.
Output: a set of valid feature assignments ω.
Generate the possible value set Ψ_{ij} *for attribute* a_{ij};
If a_{ij} *is already decided,* $\Psi_{ij} = \{defined\_value(a_{ij})\}$;
Else $x = min\_value(a_{ij})$;
    Repeat
        add x *to* Ψ_{ij};
        $x = x + step_{ij}$;
        Until $x > max\_value(a_{ij})$
Generate all possible feature assignments φ_k *based on the possible values in* Ψ_{ij};
If valid(φ_k), *add* φ_k *into* ω;
Return ω;

Given the number of negotiation issues is n, and the number of valid feature assignments is m, the complexity of Algorithm 2.4.1 is: $O(3^{\frac{n^2}{2}} * m)$.

2.4.3.2 Heuristic Search

The exponential complexity of algorithm 2.4.1 prevents it from being used for real-time applications when the number of negotiation issues and the number of valid feature assignments are large; hence a heuristic search algorithm is developed. This algorithm is based on simulating annealing search and hill climbing search. The search for a best

negotiation approach includes two parts. One is to find a best negotiation schedule; the other one is to find a best feature assignment for a given negotiation schedule.

The search for a best negotiation schedule is based on the simulated annealing idea. Given a negotiation ordering ϕ, randomly pick a POR e, if $e \in E_\phi$, remove it from E_ϕ; otherwise add it into E_ϕ. Therefore a new negotiation ordering ϕ_{new} is generated. If the negotiation schedule $\mathcal{NS}(\phi_{new})$ is better than $\mathcal{NS}(\phi)$, move to ϕ_{new}; otherwise, move to ϕ_{new} with some probability less than 1. This probability decreases exponentially with the "badness" of this move. Three heuristics have been added to this simulated annealing process:

1. Record the best negotiation schedule ever found. When the search process ends, return the best negotiation schedule ever found rather than the current one.

2. Instead of randomly deciding whether to add a POR or remove a POR, use a parameter to control the probability of the operation "add" or "remove".

3. Instead of completely randomly choosing a POR to change from current negotiation ordering, evaluate every POR e according to how the value of the negotiation schedule changes by adding this POR e to an empty POR set. The probability of adding POR e to the current POR set or removing POR e from the current POR set depends on this evaluation. A POR e with a higher positive evaluation has a higher probability of being added, and has a lower probability of being removed.

Algorithm 2.4.3 *Find the best negotiation approach.*
Input: $\mathcal{M} = (V, E)$, *the start time for negotiation* (τ), *a set of valid feature assignments* $\omega = \{\varphi_k\}, k = 1, ..., m$, *the probability to add a por:add_por_probability.*
Output:the best negotiation approach.
begin
Generate all possible $PORs = \{(v_i, v_j) | v_i, v_j \in V\}$
total_value = 0;
total_inversevalue = 0;
base_value = evaluate_schedule($\mathcal{NS}(V, \emptyset)$*,* ω*);*
for each por $\in$ *PORs*
    $\phi(por) = (V, por)$
    por.value = evaluate_schedule($\mathcal{NS}(\phi(por))$*,* ω*).value -base_value;*
    por.inverseValue = 1.0 / por.value;
    total_value += por.value;
    total_inversevalue += por.inversevalue;
for each por $\in$ *PORs*
    por.in_probability = por.value/total_value;
    por.out_probability = por.value/total_inversevalue;
best_value = minimum_value;
best_ordering = null;
best_assignment = null;
current_value = minimum_value;
current_ordering $= \emptyset$*;*

current_assignment = null;
for($t = TEMP\_MAX; t >= 0; t- = TEMP\_STEP$)
generate a random number r between [0, 1];
    if($r <$ *add_por_probability*)
        choose a por e from PORs/current_ordering according to in_probability
        new_ordering = current_ordering $\cap e$
    else
        choose a por e from current_ordering according to out_probability
        new_ordering = current_ordering - e
    evaluation_result = evaluate_schedule($\mathcal{NS}(\phi(new\_ordering)), \omega$);
    change_value = evaluation_result.value - current_value;
    if($change\_value > 0 || random < e^{-change\_value/t}$)
        current_value = evaluation_result.value;
        current_assignment = evaluation_result.assignment;
        current_ordering = new_ordering;
    if($change\_value > best\_value$)
        best_value = current_value ;
        best_assignment = current_assignment;
        best_ordering =current_ordering;
return (best_ordering, best_assignment);
end

The search for a best feature assignment is based on the hill climbing idea. Randomly pick another feature assignment φ_k. If it is better than current one, move to φ_k. After considering the characteristics of this problem, the following heuristics can be added to this search process[12]:

1. According to Algorithm 2.4.2, the change of those valid feature assignments is continuous. Based on this observation, a number of sample points with equal distance in between can be selected from all the valid feature assignments and evaluated. Hill climbing search then can be performed for each sample point which ranked in the top group[13].

2. The best feature assignment for another negotiation schedule may be used as one sample point in the search process.

Algorithm 2.4.4 *Evaluate a negotiation schedule with all possible feature assignments and find the best feature assignments and the best value.*
Input: negotiation schedule ϕ, *a set of valid feature assignments* $\omega = \{\varphi_k\}$, $k = 1, ..., m$.
Output: the best value with the best feature assignment.
begin

[12]These heuristics have not been implemented yet. Currently the search for the feature assignment process is simply a hill-climbing search process with the next selection randomly generated.

[13]The top group can be defined as top-3, top-5, or top-x, depending on how much time we want to spend, how good the approach we need to find is and other parameters in this search process.

Table 2.2: Performance of heuristic search algorithm (Number of Issues: number of negotiation issues; Approach Quality: the quality of the approach found by the heuristic search compared to the best approach found by the complete search (with quality of 1.0); Heuristic Steps: the number of search steps of the heuristic search; Complete Steps: the number of search steps of the complete search. Ratio: the ratio of heuristic search steps to complete search steps.)

Number of Issues	Approach Quality	Heuristic Steps	Complete Steps	Ratio
3	0.993	358	358	1.0
4	1.0	368	368	1.0
5	0.998	2480	5153	0.48
6	0.993	2975	71223	0.04

```
for(i=0; i<=m; i+=sample_step)
        if EV(φ, φ_i) is good enough;
            add φ_i to topset;
    best_value = minimum_value;
    best_assignment = null;
    for each φ_i in topset
        for(t=0; t<search_limit; t++)
            if(EV(φ, φ_{i+1}) > EV(φ, φ_i))
                i=i+1;
            else if(EV(φ, φ_{i-1}) > EV(φ, φ_i))
                i=i-1;
            else
                break;
        if(EV(φ, φ_i) > best_value)
            best_value = EV(φ, φ_i);
            best_assignment = i;
    return(best_value,best_assignment);
end
```

Experiments are performed to test how good this heuristic algorithm works. The limit of search steps is set as the larger number of 3000 and the number of steps for complete search. The heuristic algorithm terminates and return the best approach found when it reaches the limit. Table 2.2 shows the performance of this heuristic search algorithm. The quality of the approach it finds is very close to the best approach found by the complete search. This heuristic approach saves a large amount search effort compared to the complete search when the number of negotiation issues increases.

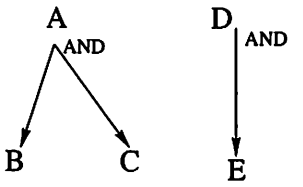


Figure 2.8: Interrelationships among negotiation issues

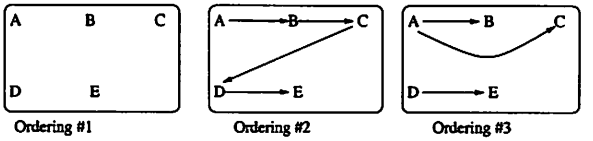


Figure 2.9: Three possible negotiation orderings

2.5 Experimental Work

2.5.1 Example

In this section, we demonstrate how the definition and the algorithm work on the supply chain examples in Figure 2.2. Figure 2.8 shows the relationships among the five issues for *Computer Producer Agent*: issue A is directly related to issue B and C, and issue D is directly related to issue E. Figure 2.9 shows three possible negotiation orderings for the five negotiation issues. Ordering #1 means all five issues are negotiated in parallel; ordering #2 means these five issues are negotiated one by one, first A, then B, then C, then D, and finally E; ordering #3 means that the negotiation on A is performed before the negotiation on B and C, and the negotiation on D is performed before the negotiation on E. Suppose the negotiation start time $\tau = 0$, and the negotiation duration on each issue is the same $\delta(v_i) = 5$, then the following negotiation schedule is generated for negotiation ordering #3 according to the assumption that every negotiation issue starts at it earliest possible time: $A[0,5]B[5,10]C[5,10]D[0,5]E[5,10]$

Besides the attributes we presented in Section 2.4.2, the following attributes need to be considered in this example:

1. time range (st, dl): the time range associated with an issue contains the start time (st) and the deadline (dl). If the issue is a task in negotiation, this task can only be performed during this range (st, dl) to produce a valid result. The larger the time range is, the higher the probability of success this negotiation issue has, since it is easier for the other agent to arrange and schedule this task. For an incoming issue, this range is already determined by the other agent who proposes this request; for an outgoing issue, the agent needs to decide what time range to propose on this issue. The agent needs to make sure this time range is consistent with all other time constraints of other issues, so it can find a feasible local schedule based on the commitment with this time range.

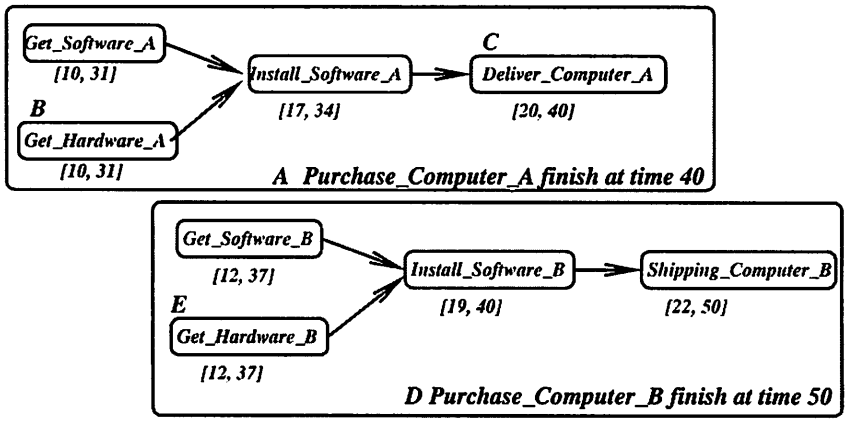


Figure 2.10: Partial order schedule

Table 2.3: Examples of negotiation approaches

case#	Issue$v$	$p_s(A)$	ϵ	$e(A)$	β	Schedule	$dl-ft$	$e(A)*(dl-ft)$	$f(v)$	$p_s(v)$
#1	A	0.9	6	0.012	22.15	A[0-3]	0	0		0.9
	B		6		1.329	B[0-4]			3.0	0.8412
	C		6		1.329	C[0-4]			0.833	0.8829
#2	A	0.92	6	0.189	1.946	A[0-3]	21	3.964		0.92
	B		6		0.117	B[0-4]			1.0	0.7817
	C		6		0.117	C[0-4]			0.5	0.8766
#3	A	0.19	9	0.117	16.52	A[0-3]	0	0		0.19
	B		9		0.991	B[3-7]			3.0	0.8412
	C		9		0.991	C[3-7]			0.667	0.8799
#4	A	0.64	9	0.006	16.56	A[4-7]	0	0		0.64
	B		9		0.993	B[0-4]			2.428	0.8289
	C		9		0.993	C[0-4]			0.667	0.8799
#5	A	0.15	13	0.043	17.68	A[0-3]	0	0		0.15
	B		13		1.060	B[3-7]			2.428	0.8289
	C		13		1.060	C[7-11]			0.833	0.8829
#6	A	0.84	11	0.142	12.58	A[8-11]	9	1.278		0.84
	B		11		0.754	B[0-4]			1.428	0.7993
	C		11		0.754	C[4-8]			1.0	0.8859

2. duration (d): if the issue is a task in negotiation, duration d is the process time requested to accomplish this task; if the issue is a resource in negotiation, duration d is the time needed for the usage of the resource.

3. flexibility (f): the flexibility is defined based on the time range and the duration: $f = \frac{dl-st-d}{d}$. The flexibility directly affects the *success probability*. For a negotiation issue with negative flexibility, the *success probability* is 0.

4. finish time (ft): the promised finish time for the task. For an incoming issue, the agent needs to decide the promised finish time (which must be no later than deadline dl the other agent requested) for this proposed task when it decides to accept this task.

5. regular reward (r): when an incoming task v is **task-level successful**, the agent's local utility increases by the amount of $r(v)$.

6. early reward rate (e): for a task in negotiation, if the contractee agent can finish the task earlier than the deadline request, it gets extra reward $e * (dl - ft)$.

The agent needs to find out how these features affect the **task-level successful reward** and the *success probability*. These relationships can be described as functions θ, ζ according to the agent's knowledge of each negotiation issue. In the negotiation process, for an incoming negotiation issue (such as A, D), the attribute needed to be decided is the promised finish time ft; for an outgoing negotiation issue (such as B, C, and E), the attributes needed to be decided are the start time (st) and the deadline (dl). The following functions describe how these attributes affect the **task-level successful reward** $\gamma(v)$ and the *success probability* $p_s(v)$.

For an incoming issue v, the **task-level successful reward** depends on the promised finish time ft: $\gamma(v) = r(v) + e(v) * (dl(v) - ft(v))$
For an outgoing issue v, the *success probability* depends on the flexibility $f(v)$, actually the time range $(st(v), dl(v))$:
$p_s(v) = p_{bs}(v) * (2/\pi) * (\arctan(f(v) + c))$ [14]
p_{bs} is the **basic success probability** of this issue v when the flexibility $f(v)$ is very large. c is a constant parameter used to adjust the relationship.

For every attribute that needs to be decided: start time (st), deadline (dl) and the promised finish time (ft), the agent can find its maximum possible range using the partial order schedule as shown in Figure 2.10. Using Algorithm 2.4.2, the agent can search over the entire possible value space, and use the partial order schedule to test if a feature assignment is valid. A set of valid feature assignments is found and sent to Algorithm 2.4.1 to find the best negotiation approach.

To make the output easier to understand, only issues A, B and C are considered in the following example. Table 2.3 shows the output of Algorithm 2.4.1 on the example in Figure 2.8 given following parameters:

- regular reward $r(A) = 19$;

[14] Obviously this function could be affected by the meta-level information from the other agent.

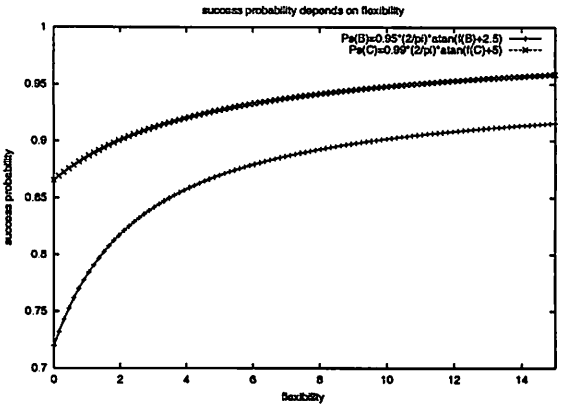


Figure 2.11: Success probability depends on flexibility

- negotiation duration $\delta(A) = 3$;
- negotiation duration $\delta(B) = 4$;
- negotiation duration $\delta(C) = 4$;
- $p_{bs}(B) = 0.95$;
- $p_s(B) = p_{bs}(B) * (2/\pi) * (\arctan(f(B) + 2.5)))$;
- $p_{bs}(C) = 0.99$;
- $p_s(C) = p_{bs}(C) * (2/\pi) * (\arctan(f(C) + 5)))$.

The different constant parameters for $p_s(B)$ and $p_s(C)$ specify that issue C has a higher *success probability* than issue B given the same flexibility, as shown in Figure 2.11.

The following parameters are randomly generated:

1. the *success probability* of A, $p_s(A)$;
2. negotiation deadline ϵ ($\epsilon(A) = \epsilon(B) = \epsilon(C)$);
3. early reward rate $e(A)$;
4. decommitment penalty $\beta(A)$, $\beta(B)$, and $\beta(C)$.

In both case 1 and case 2, the negotiation deadline $\epsilon = 6$, so the valid negotiation ordering has the three negotiation issues performed in parallel. In case 2, issue A has a higher earlier reward rate $e(A)$, and all issues have lower decommitment penalties β than in case 1, so the negotiation approach in case 2 arranges task A to finish 21 time units earlier than the requested deadline, and earns an extra reward of 3.964. In exchange, issues B and C have smaller flexibilities $f(B)$ and $f(C)$, hence lower success probability $p_s(B)$ and $p_s(C)$. In case 3 and case 4, the negotiation deadline $\epsilon = 9$. In case 3, issue A has a much lower success probability $p_s(A)$ than in case 4, so the negotiation on A is scheduled before the negotiation on B and C. In case 5 and case 6, the negotiation deadline $\epsilon = 11$ and

the negotiation issues on A, B and C are sequenced according to the success probabilities; the issues with lower *success probability* start earlier. In case 6, issue A has a higher earlier reward rate $e(A)$, and all issues have lower decommitment penalties β than case 5, so the negotiation approach in case 6 arranges task A to finish 9 time units earlier than the requested deadline; this earns an extra of reward 1.278. In exchange, issues B and C have smaller flexibilities $f(B)$ and $f(C)$ and hence lower success probabilities $p_s(B)$ and $p_s(C)$. It is also important to notice that in all cases, issue B gets larger flexibility than issue C, but has a similar *success probability* to that of issue B. This occurs because it is much easier for issue C to achieve a successful negotiation according to the function that defines the relationship between the *success probability* and the flexibility.

2.5.2 Experiments

We have implemented the search and evaluation algorithms so as to enable the reasoning in the multi-linked negotiation process as indicated in the examples described in Section 2.5.1. We designed the following experiments to study how the different negotiation approaches affect the agent's performance. The experimental environment was set up based on the scenario described in Section 2.1.1. Four agents were built using the JAF agent framework [62]. New tasks were randomly generated with decommitment penalty $\beta \in [0, 25]$, early finish reward rate $e \in [0, 0.2]$, and deadline $dl \in [60, 70]$, and arrived at the contractee agents periodically. We use the same task structures as described in Figure 2.6, tasks vary with randomly generated parameters. This scenario represents a class of problem where one agent needs to deal with both directly and indirectly related negotiation problems. The deadlines of tasks are randomly generated from a range which allow the agent to choose different negotiation orderings.

In this experiment, *Computer Producer Agent* needs to deal with the multi-linked negotiation issues related to the incoming task *Purchase_Computer* and the outgoing task *Get_Hardware* and *Deliver_Computer*. The following three different negotiation approaches were tested:

1. Sequenced Negotiation. The agent deals with the negotiation issues one by one, first the outgoing negotiation issues, then the incoming negotiation issues. The finish time promised is the same as the deadline requested from the other agent, and the outgoing issues get the largest possible flexibilities.

2. Parallel Negotiation. The agent deals with the negotiation issues in parallel. It arranges reasonable flexibility (1.5, in this experiment) for each outgoing task, and based on this arrangement, the finish time of the incoming task is decided and promised to the contractee agent.

3. Decision-Based Negotiation. The agent deals with the negotiation as the best negotiation approach suggests. The best negotiation approach comes from the decision-making process using the search algorithm 2.4.1.

The entire experiment contains 40 group experiments. Each group experiment has the system running for 1000 time clicks for three times and each time the *Computer Producer Agent* uses one of the three different approaches.

Table 2.4: Comparison of performance using different negotiation approaches

Policy	Tasks Received	Task Accepted	Task Canceled	Decommit Penalty	Early Reward	Utility
Sequenced Negotiation	60	60	37.25	73.82	0	358.09
Std.Dev.	0	0	2.6	11.8	0	57.4
Parallel Negotiation	60	60	23.70	333.20	29.06	385.20
Std.Dev.	0	0	2.6	47.6	17.0	86.8
Decision-Based Negotiation	60	60	25.78	56.65	185.79	779.16
Std.Dev.	0	0	2.4	23.5	47.8	62.3

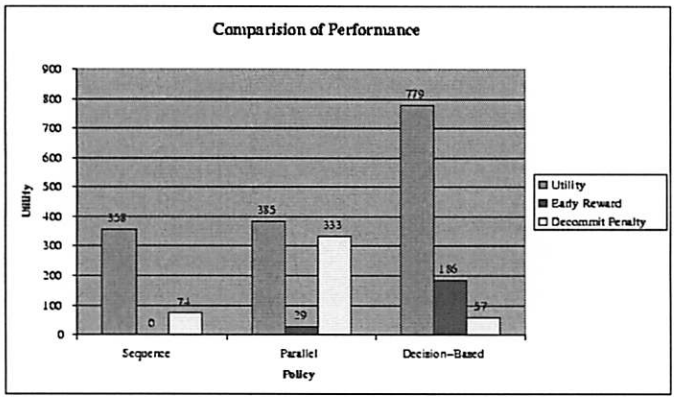


Figure 2.12: Comparison of performance using different negotiation approaches in graph

Table 2.4 shows the comparison of the *Computer Producer Agent*'s performance using different strategies. When the agent uses the sequenced negotiation approach, more tasks are canceled because of the missed negotiation deadlines. When the agent uses the parallel negotiation approach, the agent pays a higher decommitment penalty because the failure of the sub-contracted task prevents the incoming task to be task-level successful. The decision-based approach is obviously better than the other two approaches[15]. It chooses to have the negotiation approach dynamically according to the negotiation deadline and other attributes. Under this experimental setup, it chooses the case where all negotiations are performed in parallel about 13% of the time; it chooses the case where all negotiations performed sequentially about 38% of the time, and the other times it chooses the case where some negotiations are performed in parallel. This strategy enables the agent to get more early reward and pays fewer decommitment penalties. Figure 2.12 shows the utility, early reward, and decommitment penalty of *Computer Producer Agent* using three different negotiation approaches.

The experimental result shows that in a multi-linked negotiation situation, it is very important for the agent to reason about the relationship among different negotiation issues and make a reasonable decision about how to perform negotiation. This decreases the likelihood of the need for decommitment from previously settled issues and increases the likelihood of utility gain.

2.6 Summary

In this chapter, we discussed what the multi-linked negotiation problem is and how an agent should deal with the multi-linked negotiation problem. Multi-linked negotiation deals with multiple negotiation issues. When these issues are interconnected, the negotiation over one issue has influence on the negotiations over other issues. To solve a multi-linked negotiation problem, the agent needs to find out in what order the negotiations should be performed, and how to negotiate on each issue to avoid conflict among them. A partial oder schedule is built, which allows the agent to reason about the time-related constraints and the flexibilities on each issue. This reasoning process is important for the agent to perform conflict-free negotiation and manage flexibilities in negotiation.

Furthermore, a formalized model of the multi-linked negotiation problem enables the agent to represent and reason about the relationships among different negotiation issues explicitly. Using this model, the best negotiation approach can be found by a complete search algorithm. A heuristic search algorithm finds the nearly-optimal approach in affordable time. Experimental work shows that this management technique for multi-linked negotiation leads to improved performance.

[15]Using a t-test, with the 0.001 alpha-level, the following hypothesis H_o is rejected: when using the decision-based approach, *Computer Producer Agent* achieves an extra utility that is equal to 100% of the utility gained when using the sequenced negotiation approach, and 78% of the utility gained when using parallel negotiation approach, compared to the hypothesis H_a: when using the decision-based approach, *Computer Producer Agent* achieves an extra utility that is more than 100% of the utility gained when using the sequenced negotiation approach, and 78% of the utility gained when using parallel negotiation approach.

CHAPTER 3

INTEGRATIVE NEGOTIATION

In this chapter we will address the problem of negotiation in a complex organizational context. First we will discuss what the problem is and why it is important (Section 3.1); then we will briefly introduce the MQ framework, which acts as a fundamental support for this work (Section 3.2). In Section 3.3 we will describe the integrative negotiation mechanism in detail. An example is used to explain how this mechanism works in Section 3.4. Experimental work is presented in Section 3.5, which shows how the different cooperative strategies affect the agents' performance and social welfare. Section 3.7 summarizes this chapter.

3.1 Between Self-Interested and Completely Cooperative

Negotiation research falls into two broad classes: cooperative negotiation and competitive negotiation. In competitive negotiation, agents are said to be *self-interested* and negotiate to maximize their own local utility. In cooperative negotiation, agents work to find a solution that increases their joint utility – the sum of the utilities of all involved agents. Significant work [48, 51] has been done in the area of bounded rational *self-interested* agents (BRSI), which engage in competitive negotiation. Said agents are *self-interested*, and social welfare is not a concern – each agent works to maximize its own utility through contracting, bidding and decommiting. Significant work has also been done in the area of conflict resolution through cooperative negotiation [15, 30, 59]. In such environments, there is no notion of individual agent utility – agents are *completely cooperative* with each other and work together to solve problems with the goal of maximizing social welfare.

We feel that as the sophistication of multi-agent systems increases, MAS will be neither simple market systems where each agent is purely *self-interested*, seeking to maximize its local utility, nor distributed problem solving systems where all agents are *completely cooperative*, working to maximize the utility derived from a set of global goals. This type of hybrid system will occur for the following reasons. One reason is that agents from different and separate organizational entities will come together to dynamically form virtual organizations or teams to solve specific problems that are relevant to each of their organizational entities[40]. How these agents work as a team will often be dependent on the existence of both long-term and short-term relationships and on the confrontational attitude of their underlying organizational entities. We also feel that even for agents from *self-interested* organizations, it might be beneficial for them to be partially cooperative when they are in situations where they will have repeated transactions with agents from other organizational

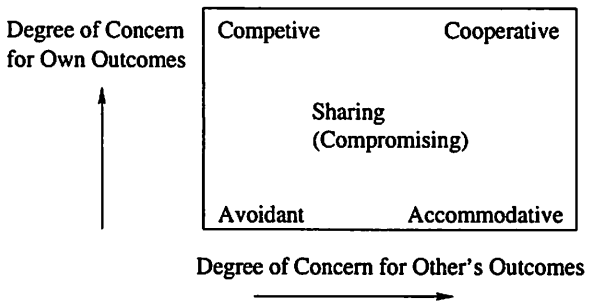


Figure 3.1: The dual concern model

entities. Additionally, agents may be involved concurrently with more than one virtual organization while doing tasks for their own organizational entity. Furthermore, we feel that even for agents working solely with agents of their own organizational entity, it may still be advantageous for them to take varying attitudes from fully cooperative to purely *self-interested* for the organization to best achieve its overall goals. This perspective is based on a bounded-rational argument: it is not practical from a computational or communication perspective for an agent to be fully cooperative, because a fully cooperative agent would need to take into account the utilities of all agents in the organization, as well as the state of achievement of all organizational goals. Thus, it is our feeling that it may be best for the organization to have agents that are only partially cooperative in their local negotiation with other agents to more effectively deal with the uncertainty of not having a completely informed and up-to-date view of the state of the entire agent organization.

In such an environment, multi-agent systems will thus consist of large groups of loosely coupled agents that work together on tasks. The relationships among agents will depend on their organizational roles and may be of any type from purely *self-interested* to totally cooperative. This is the complex organizational problem space the MQ [67, 66] framework is designed to represent. Note that this work pertains to complex agents. We assume that *Agents* are autonomous, computing entities that have the ability to choose which tasks to perform and when to perform them. Agents are also rationally bounded, resource bounded, and have limited knowledge of other agents. Agents can perform tasks locally if they have sufficient resources and they may interact with other agents. The agents will have choices about with whom to collaborate, how to negotiate, what to charge for services, etc. Further, the negotiation strategy will be dependent on the relationships among the negotiating parties and the particular negotiation issue.

We therefore feel that in a complex agent society, an agent will need to work with other agents from a variety of different organizational positions. For example, it may interact with an agent from its own group, an agent who has a higher position and thus more authority, an agent from a cooperative company, or an agent from a competing company and so forth. The agent's attitude toward negotiation is not simply either competitive or cooperative; the agent needs to qualitatively reason about each negotiation session, e.g., how important its own outcome is compared to the other agents' outcomes, so that it can choose an appropriate negotiation strategy.

Figure 3.1 describes this dual concern model [34]. When the agent only attaches importance to its own outcome, its attitude toward negotiation is competitive (self-interested); when an agent attaches the same degree of importance to its own outcome as it does to the outcomes of the other agent, its attitude is cooperative; when the agent attaches more importance to the outcomes of other agents and no importance to its own outcome, its attitude is accommodative; if the agent attaches no importance to any outcomes, its attitude is avoidant (negotiation is not worth its time and effort). From this model, we find that there are potentially many options between the two extremes of *self-interested* and cooperative. These other options depend on the importance the agent attaches to the increase of its own utility relative to the importance it attaches to the other agents' utility increases. Later in this chapter, we present an integrative mechanism that enables an agent to qualitatively manage its attitude towards each negotiation session. This mechanism is not purely *self-interested* or purely cooperative, but supports ranges of these behaviors so that the agent can reason about how cooperative it should be. This mechanism is based on the Motivational Quantities (MQ) framework [67, 66], which is introduced in Section 3.2.

3.2 MQ Framework

The MQ framework[1] is an agent control framework that provides the agent with the ability to reason about which tasks should be performed and when to perform them. The reasoning is based on the agent's organizational concerns.

The basic assumption is that agents are complex, with multiple goals related to the multiple roles they play in the agent society. The progress toward one goal can not substitute for the progress toward another goal. Motivational (MQs) are used to represent the progress toward organizational goals quantitatively.

Each agent has a set of MQs which it is interested in and wants to accumulate. Each MQ_i in this set represents the progress toward one of the agent's organizational goals. Each MQ_i is associated with a preference function (utility curve), U_{f_i}, that describes the agent's preference for a particular quantity of the MQ_i. Different types of MQs are not interchangeable. The utility U_i associated with MQ_i can not be transferred to the utility U_j associated with MQ_j if $i \neq j$.

The agent's overall utility is a function of the different utilities associated with the MQs it tracks: $U_{agent} = \gamma(U_i, U_j, U_k, ..)$. The structure of function γ represents the agent's preference and emphasis on different organizational goals. The MQ framework thus provides an approach to compare the agent's different motivational factors through a multi-attribute function.

Not all agents have the same MQ set. If two agents need to construct a commitment through coordination or negotiation, and use MQ as an exchange medium, they need to have at least one MQ in common, or be willing to form one dynamically. Different agents may have different preferences for the same MQ.

MQs are consumed and produced by performing of MQ tasks. The agent's ultimate goal is to select tasks to perform in order to maximize its local utility achievement through

[1]The description of MQ framework in this section is based on Wagner's thesis work. The same notation is used here to maintain consistency. More information is available from [67, 66].

collecting different MQs. This does not mean that the agent has to be "self-interested"; it only means that the agent selects its actions to contribute to its multiple organization goals. If "to help agent B" is one of the goals of agent A, then agent A will act as a cooperative to agent B. If two or more agents have a goal in common and hence have the same MQ in common, they act as a group or a team working collaborately toward this goal.

MQ tasks are an abstraction of the primitive actions that an agent may perform. Each MQ task is associated with one or more organizational goals, and the performing of this task produces progress (MQs) toward these goals. The agent compares and selects tasks that are associated with different organizational goals.

Each MQ task T_i has the following characteristics:

- Earliest start time, $start_i$. The performance of T_i before this time does not generate valid results.
- Deadline, $deadline_i$. The accomplishment of T_i after this time does not generate valid results.
- A set of MQ alternatives. Each alternative specifies a different performance profile of this task. Each alternative has the following specifications:
  - it needs some process time to be accomplished, denoted as d_i.
  - it produces certain quantities of one or more MQs, denoted as $MQPS$ (MQ production set). The production of MQs reflects the progress made by the accomplish of the task toward a goal.
  - it consumes certain quantities of one or more types of MQs, denoted as $MQCS$ (MQ consumption set). The consumption of MQs represents resources consumed by performing this task, or favors owed to other agents for subcontracting work.

The MQ scheduler schedules current potential MQ tasks and their alternatives, and produces a schedule of a set of MQ tasks, specifying their start times, finish times and which alternative is chosen for each task. The scheduler takes the following factors into consideration: the $MQPS$, $MQCS$, duration d_i for each MQ alternative, the earliest start time and the deadline of each MQ task.

The MQ framework provides the comparison of tasks that need to be performed for different reasons: for different organizational goals, for other agents to gain some financial benefit or favors in return, for cooperation with other agents, etc. It also supports different utility functions that relate the execution of tasks to the importance of organizational goals.

3.3 Integrative Negotiation

The motivational quantities (MQ) [67, 66] framework provides an agent with the capability to reason about different goals in an open, dynamic and large-scale MAS, allowing the agent to evaluate a negotiation issue from an organizational perspective. The MQ framework quantifies different underlying motivational factors and provides the means to

compare them via a multi-attribute utility function. This framework can therefore support sophisticated negotiation where each negotiation issue has MQ transference associated with it. Let's use task allocation as an example of negotiation, where for each task t allocated to agent B, from agent A, certain MQs are transferred from agent A to agent B. The conceptual model here is that agent B is motivated by the potential increase in its MQs to perform tasks for agent A (note that this does not convert the MQs to currency as not all agents may be interested in said MQs). We will start with a simple, abstract example. In this model, when agent B commits to accomplishing task t, based on a contract that is mutually agreed upon by the two agents (formed either dynamically or pre-defined), it is then obligated to perform the task. When B successfully accomplishes t, the agreed upon amount of the MQ will be transferred from agent A to agent B. Note that agent B must actually decide whether or not it is interested in performing t. This evaluation is done via the MQ framework and the associated MQ scheduler. The evaluation uses agent B's preference for the MQ in question to determine the the relative value of performing t for agent A. This valuation process, in turn, determines agent B's attitude toward the negotiation of task t.

In terms of specifics, there are two types of MQs that could be transferred with the successful accomplishment of task t: *goal_related* MQ and *relational* MQ. These classes are conceptual and only used to clearly differentiate motivations for task performance from attitudes toward negotiation issues – in reality, they are both simply MQs. *Goal_related* MQs are associated with an agent's organizational goals; increases in MQ volume generally have positive benefits to the agent's utility. Note that the agent's designer determines which kinds of MQs the agent tracks (and is interested in), defines the agent's preference for each via the utility functions discussed earlier, and determines how these relate to the agent's organizational goals. In this work, we will assume that they do not change during the agent's life to simplify the experiments. When dealing with *goal_related* MQs, the agent collects MQs for its own utility increase. In this sense, agent B's performance of task t is motivated by "*self-interested*" reasons if payment is via a *goal_related* MQ. As an example, consider task t which has 3 units of MQ_x transferred with it, and an agent B with a MQ_x utility curve of $utility(x) = 2x$ that means, the utility of agent B will increase by 6 units by collecting 3 units of MQ_x though performing task t. Agent B decides whether to accept task t by reasoning about this value relative to the cost of the resources it will expend in the performance of t. In this case, as the task doesn't consume any MQs, the resource expenditure is time or in terms of opportunity cost. Because this reasoning process pertains to *goal_related* MQs, it is "*self-interested*", for the agent's only concern is its own utility increase.

Consider a related case. Suppose that by having task t accomplished, agent A's own utility increases by 20 units. If agent B takes this fact into favorable consideration when it makes its decision about task t, then agent B will be cooperative with agent A because B is also concerned about A's outcome (in addition to its own). If we want B to consider A's utility, we need to introduce another MQ designed to model B's (revised) preference for A to have a utility increase also. To reflect B's attitude toward A's outcome, we introduce a *relational* MQ, the preference for which represents how cooperative B is with A concerning task t. Let $MQ_{ba/t}$ be the relational MQ transferred from agent A to agent B when agent B performs task t for agent A. Since $MQ_{ba/t}$ is a relational MQ, its only purpose

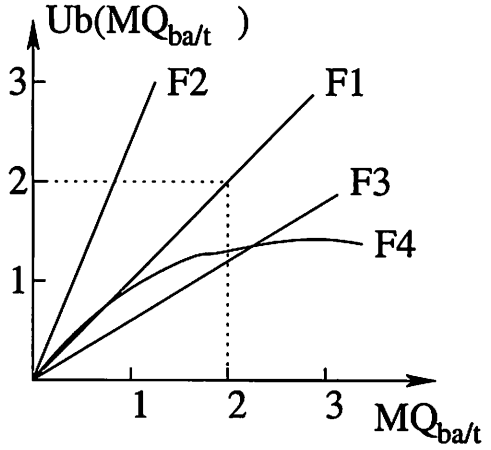


Figure 3.2: Different mapping functions of $MQ_{ba/t}$

is to measure the relationship between agents A and B. While agent B may actually have an organizational goal to accumulate MQs of this type[2], in this thesis, for simplicity of presentation, we will assume that agent B does not have an organizational level goal to cooperate with agent A. Accordingly, when measuring the utility of agent B toward problem solving, we will not consider the utility produced by any relational MQs such as $MQ_{ba/t}$. Likewise with agent A, when agent A transfers $MQ_{ba/t}$ to agent B, we will not tabulate the negative change in utility of agent A because the change in utility is not related to problem solving progress, but is instead related to the transfer of a relational MQ. The reason for this approach is that in this thesis our performance metric is social welfare as it is conventionally used, which is in terms of progress toward joint goals. From this view, the utility produced by a relational MQ can be seen as *virtual utility*. Though $MQ_{ba/t}$ produces virtual utility, is important because it carries the information of how important task t is for agent A[3] and makes it possible for agent B to consider agent A's outcome when it makes its own decisions. In practice, how $MQ_{ba/t}$ is mapped into agent B's (virtual) utility, meaning utility that is not included in the social welfare computation[4], depends on how cooperative agent B is with agent A. Suppose that 20 units of $MQ_{ba/t}$ are transferred with task t to agent B, representing the utility agent A gained by having agent B perform task t, Figure 3.2 shows four different functions for mapping $MQ_{ba/t}$ to agent B's utility.

Function F1, F2 and F3 are linear functions: $U_a(MQ_{ba/t}) = k * MQ_{ba/t}$.

[2]In this case, the agent's local utility would also increase by accumulating MQs of this type, as an indication that cooperating with the other agent fosters its organizational objective.

[3]It is assumed that agents are honest and don't lie about the importance of task t.

[4]In remainder of the thesis, we may omit the word "virtual" before utility, but we know that this *relational* MQ only maps into virtual utility that is not real utility. In the experimental work, neither the agent's utility nor the social welfare includes the virtual utility from *relational* MQ.

If $k = 1$ (F1), $U_b(MQ_{ba/t}) = MQ_{ba/t} = U_a(t)$ ($U_a(t)$ denotes the utility agent A gained by transferring t), then agent B is completely cooperative to agent A[5];

If $k > 1$ (F2), $U_b(MQ_{ba/t}) > MQ_{ba/t} = U_a(t)$, then agent B is accommodative to agent A;

If $k < 1$ (F3), $U_b(MQ_{ba/t}) < MQ_{ba/t} = U_a(t)$, then agent B is partially cooperative with agent A;

If $k = 0$, $U_b(MQ_{ba/t}) = 0$, then agent B is *self-interested* with respect to agent A. In this case, if agent A wants agent B to do t, it needs to transfer another kind of MQ (the *goal_related* MQ) to agent B, agent B and agent A can negotiate about what type of *goal_related* MQ to transfer and how much of it should be transferred, regarding how and when agent B could accomplish task t. In the following examples and the experimental work, we assume that the type and amount of the transferred *goal_related* MQs are fixed and agents do not negotiate about them, so we can demonstrate how the *relational* MQ works.

The mapping function could also be a nonlinear function (F4) that describes a more complicated attitude of agent B to agent A, i.e., agent B being fully cooperative with agent A for some period and then becoming *self-interested.* An agent can adjust the utility mapping function to reflect its relationship with another agent, which could be it's administrator, colleague, friend, client or competitor. By adjusting some parameters in the mapping function, more subtle relationships could be managed. The agent could differentiate a friendly colleague from an unfriendly colleague; also it could draw distinctions between a best friend and an ordinary friend.

Different from the *goal_related* MQs, which are built by the agent's designer and whose utility curves are not changing, the utility curves of the *relational* MQs can be adjusted by the agent dynamically to reflect its dynamic relationships with other agents. The agent's attitude towards another agent could be "issue-specific"; given that an agent can play multiple roles, there may be different issues negotiated between agents, and the agents should select an appropriate attitude according to what issue is negotiated. For example, for a colleague's request to contribute to a shared professional job, and for the same colleague's request for a ride to the airport, even though both requests come from the same agent, the agent's attitude could be different.

By introducing this agent-oriented, issue-specific *relational* MQ into negotiation, the agent's attitude toward another agent concerning a specific issue can be represented as the utility curve associated with the *relational* MQ. This mechanism is called an *integrative negotiation* mechanism, which supports the agent's choosing a negotiation attitude of any type from from purely *self-interested* to totally cooperative.

How can an agent choose its attitude toward other agents in such a complex organization context? We are not planning to present a detailed solution to this question in this thesis, but we feel that the agent should dynamically adjust its attitude by analyzing the other party, the issue in negotiation and its current problem-solving status. The following information should be considered in this decision making process: "Who is the other agent?", "How are

[5]It should be noticed that the relationship between agents is not symmetric; the fact that agent B is *completely cooperative* to agent A does not imply that agent A is also *completely cooperative* to agent B.

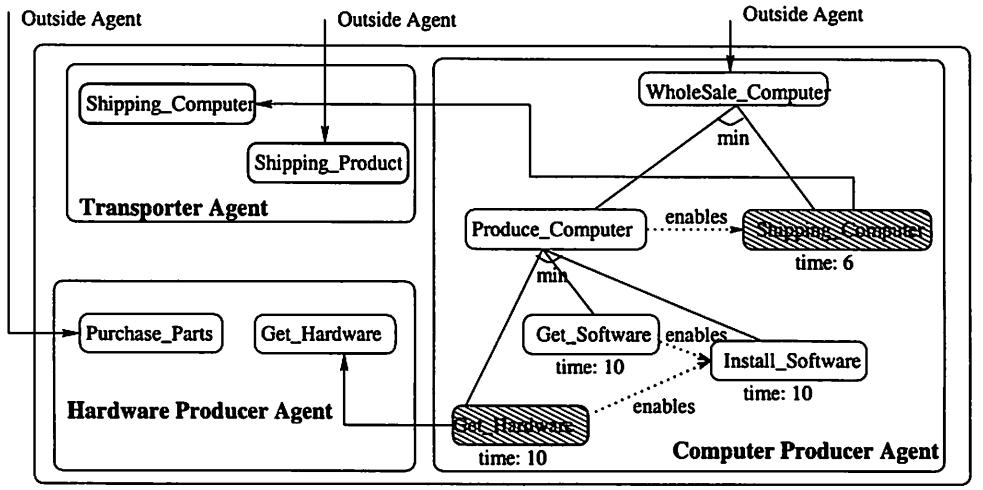


Figure 3.3: Agent society of three agents

its organizational goals related to mine?", "What is its objective?", "What is its relationship to me?" and so forth. Some of this information can be learned from experience [56].

In the MQ framework, the MQ scheduler enables the agent to optimize its schedule and maximize its local utility. While the framework directly supports the concept of relational MQs and being motivated to cooperate on that basis, the use of MQ transference in this thesis extends the MQ framework to interconnect the local scheduling problems of two or more agents in a dynamic fashion (based on the current context). Prior to this work, no meaningful research has studied MQ transference or its implications.

In Section 3.4, we will introduce an example of a three-agent society and show how the integrative negotiation mechanism works using the MQ framework.

3.4 The Scenario

There are three agents in this society as shown in Figure 3.3:

1. *Computer Producer Agent* (*c*): receives the *Produce_Computer* task from an outside agent (which is not considered in this example). Figure 3.3 shows that to accomplish *Produce_Computer* task, *Computer Producer Agent* needs to generate an external request for hardware (*Get_Hardware* task), and also needs to ship the computer (*Shipping_Computer*) through a transport agent.

2. *Hardware Producer Agent* (*h*): receives task *Get_Hardware* from *Computer Producer Agent*; it also receives task *Purchase_Parts* from an outside agent.

3. *Transporter Agent* (*t*): receives task *Shipping_Computer* from *Computer Producer Agent*, it also receives task *Shipping_Product* from an outside agent.

In this simple example, every agent collects the same type of *goal_related* MQ: $MQ_{\$}$. The utility curve for $MQ_{\$}$ is: $utility(x) = x$, and every agent uses this same function. Each task that the agent receives includes the following information:

- deadline (dl): the latest finish time for the task.
- reward (r): If the task is finished by the deadline, the agent will get reward r (which is r units of $MQ_\$$).
- early finish reward rate (e): If the agent can finish the task by the time (ft) as it promised in the contract, it will get the extra early finish reward: $max(e * r * (dl - ft), r)$[6] in addition to the normal reward r.

Hardware Producer Agent receives *Purchase_Parts* task from an outside agent with x units of $MQ_\$$, where x is a random number varying from 2 to 10. *Computer Producer Agent* has a long-term contract relationship with *Hardware Producer Agent* and *Transporter Agent*: its *Get_Hardware* task always goes to *Hardware Producer Agent* with a fixed reward of 3 units of $MQ_\$$, and its *Shipping_Product* task always goes to *Transporter Agent* with a fixed reward of 3 units of $MQ_\$$. Every *Produce_Computer* task comes to *Computer Producer Agent* with a reward of 20 units of $MQ_\$$. If it is finished by its deadline, *Computer Producer Agent* would have its local utility increased by 14 units. Assuming tasks *Get_Hardware* and *Shipping_Product* have the same importance, the accomplishment of each task would result in a utility increase of 7 units for *Computer Producer Agent*. This information is reflected by the 7 units of $MQ_{hc/t}$ transferred with task *Get_Hardware* and 7 units of $MQ_{tc/t}$ transferred with task *Shipping_Product*. $MQ_{hc/t}$[7] is a *relational MQ* introduced to reflect the relationship of *Hardware Producer Agent* with *Computer Producer Agent* concerning task t. The $MQ_{hc/t}$ transferred with the task represents the utility increase of *Computer Producer Agent* by having this task accomplished. How it is mapped into *Hardware Producer Agent*'s virtual utility depends on *Hardware Producer Agent*'s attitude towards the utility increase of *Computer Producer Agent* regarding task *Get_Hardware*. If the *Produce_Computer* task could be finished earlier than its deadline, *Computer Producer Agent* could get more than 20 units of reward. The extra utility increase could be estimated and reflected by more than 7 units of $MQ_{hc/t}$ or $MQ_{tc/t}$ being transferred to the other two agents. Suppose the following task is received by *Computer Producer Agent*:

task name : Purchase_Computer_A
earliest start time: 10
deadline: 70
reward: 20 units $MQ_\$$
early finish reward rate: e=0.01

Through the reasoning of the MQ scheduler, *Computer Producer Agent* decides to accept the task and finish it by time 40 (this leaves 4 units of slack time) to earn an extra early reward of 6 ($(70 - 40) * 0.01 * 20$) units of $MQ_\$$. Its local utility increases by 20 units after the accomplishment of this task. Hence, the following two task requests are sent to *Hardware Producer Agent* and *Transporter Agent* respectively:

[6]For each time unit the task finishes earlier than the deadline, the contractee agent gets extra reward $e * r$, but the total extra reward would exceed the reward r.

[7]Similarly, $MQ_{tc/t}$ is a *relational MQ* that reflects the relationship of *Transporter Agent* with *Computer Producer Agent* concerning task t. Detailed discussion about it is omitted here.

task name : Get_Hardware_A
earliest start time: 10
deadline: 20
reward: 3 units $MQ_{\$}$, *10 units* $MQ_{hc/t}$
early finish reward rate: e=0.01[8]
task name : Shipping_Computer_A
earliest start time: 30
deadline: 40
reward: 3 units $MQ_{\$}$, *10 units* $MQ_{tc/t}$
early finish reward rate: e=0.01

In this example, we look at three different attitudes based on the linear function: $U_{ha}(MQ_{hc/t}) = k * MQ_{hc/t}$.

1. k=1, *Hardware Producer Agent* is *completely cooperative* to *Computer Producer Agent* regarding task *Get_Hardware*.

2. k=0.5, *Hardware Producer Agent* is half-cooperative (partially cooperative) to *Computer Producer Agent* regarding task *Get_Hardware*.

3. k=0, *Hardware Producer Agent* is *self-interested* to *Computer Producer Agent* regarding task *Get_Hardware*.

Now we can look at how these different attitudes affect the negotiation process of *Hardware Producer Agent*. Suppose there are two other tasks *Purchase_Parts_A* and *Purchase_Parts_B* received by *Hardware Producer Agent* besides task *Get_Hardware_A*. The following three tasks are then sent to the MQ Scheduler (suppose the initial MQ value is $\emptyset$):

task name : Get_Hardware_A
earliest start time: 10
deadline: 20
process time: 10
MQPS: $[MQ_{\$},3]$, $[MQ_{hc/t}, 10]$
task name : Purchase_Parts_A
earliest start time: 10
deadline: 30
process time: 10
MQPS: $[MQ_{\$},4]$
task name : Purchase_Parts_B
earliest start time: 10
deadline: 20
process time: 10
MQPS: $[MQ_{\$},9]$

[8]Assume *Computer Producer Agent* assigns the same early finish reward rate to this task as the task *Produce_Computer* it receives.

If *Hardware Producer Agent* is *completely cooperative* to *Computer Producer Agent*, the best MQ schedule produced is:

Agenda 3.4.1 $[10, 20]Get\_Hardware\_A$ $[20, 30]Purchase\_Parts\_A$

Hardware Producer Agent will have 7 units utility increase after the accomplishment of this schedule. If *Hardware Producer Agent* is *self-interested* to *Computer Producer Agent*, the best MQ schedule produced is:

Agenda 3.4.2 $[10, 20]Purchase\_Parts\_B$ $[20, 30]Purchase\_Parts\_A$

Hardware Producer Agent will have 13 units utility increase after the accomplishment of this schedule. If *Hardware Producer Agent* is half-cooperative to *Computer Producer Agent*, the best MQ schedule produced is the same as above. However, if task Purchase_Parts_B comes with 7 units $MQ_{\$}$ instead of 9 units, the best MQ schedule produced is:

Agenda 3.4.3 $[10, 20]Get\_Hardware\_A$ $[20, 30]Purchase\_Parts\_A$

Hardware Producer Agent will have 7 units utility increase after the accomplishment of this schedule.

A similar reasoning process also applies to the *Transporter Agent*. The above example shows how an agent's reaction in a negotiation process depends on its attitude towards the other agent regarding the issue in negotiation, and also is affected by the other tasks on its agenda. The more cooperative an agent is, the more it will sacrifice its own utility for the other agent's utility increase. This integrative negotiation mechanism enables the agent to manage and reason about different cooperative attitudes it could have with another agent regarding a certain issue.

3.5 Experimental Work

The example in Section 3.4 shows that an agent needs to sacrifice some of its own utility to be cooperative with another agent. The question is: Could cooperative agents increase the social welfare[9]? Is it always true that a cooperative strategy can improve the social welfare? When should an agent be cooperative and how cooperative should it be?

To explore these questions, the following experimental work was done based on the scenario described in Section 3.4[10]. This scenario represents a class of situations where the accomplishment of one task requires more than one agent's cooperation. *Hardware Producer Agent* has a choice of three different attitudes toward *Computer Producer Agent*: *completely cooperative* (C), *half-cooperative* (H), and *self-interested* (S); *Transporter Agent* has

[9]Social welfare refers to the sum of the utilities of all the agents in the society which is considered, i.e. in above example, the social welfare is the sum of the utilities of the three agents: *Computer Producer Agent*, *Hardware Producer Agent*, and *Transporter Agent*.

[10]The experiments are performed in the MASS simulator environment[22], and the agents were built using the JAF agent framework[62].

Table 3.1: Comparison of performance under different cooperative situations

	Utility of Computer Producer Agent	ratio	Utility of Hardware Producer Agent	ratio	Utility of Transport Agent	ratio	Social Welfare	ratio
SS	218	1.000	575	1.000	856	1.000	1649	1.000
CC	842	4.08	415	0.72	766	0.90	2022	1.23
HH	587	2.84	493	0.86	806	0.94	1886	1.14
SC	301	1.41	587	1.02	798	0.93	1686	1.02
CS	469	2.24	364	0.63	839	0.98	1672	1.01
HS	390	1.87	467	0.81	845	0.99	1702	1.03
SH	292	1.36	585	1.02	815	0.95	1692	1.03
HC	632	3.06	500	0.87	772	0.90	1905	1.16
CH	761	3.68	405	0.70	802	0.94	1967	1.19

Table 3.2: Results from statistical tests

Object	Number to Compare	Ho	Ha	Result	Alpha	p
CC - SS	330	=330	> 330	Reject Ho	0.01	0.008
HH - SS	180	=180	>180	Reject Ho	0.01	0.0008
SC - SS	0	=0	>0	Fail to reject Ho	0.01	0.0179
CS - SS	0	=0	>0	Fail to reject Ho	0.01	0.0965

the same three choices, so there are 9 combinations: SS (both agents are *self-interested*), SC (*Hardware Producer Agent* is *self-interested* while *Transporter Agent* is *completely cooperative*), SH (*Hardware Producer Agent* is *self-interested* while *Transporter Agent* is *half-cooperative*), HS, HC, HH, CS, CH, CC. The data is collected over 48 groups of experiments. Each group experiment runs 9 times for 1000 time clicks each, and each time the agents work on the same incoming task set under one of the nine different situations. The tasks in each set for each group experiment are randomly generated with different rewards, deadlines and early reward rates within certain ranges

Table 3.1 shows the comparison of each agent's utility and the social welfare under these different situations. The percentage numbers are the normalized utility numbers based on the utility gained when the agent is *self-interested.* Table 3.1 shows that when both *Hardware Producer Agent* and *Transporter Agent* are *completely cooperative* to *Computer Producer Agent* (CC), the society gains the most social welfare. Even when both agents are only *half-cooperative* (HH), the social welfare is still very good. However, when one agent is *completely cooperative*, and the other agent is *self-interested* (CS, SC), the social welfare does not improve much compared to the completely self-interested (SS) case. The reason for the lack of significant improvement is that, in this example, to accomplish task *Produce_Computer* requires both task *Get_Hardware* and task *Shipping_Computer* to

Table 3.3: Utility of *Hardware Producer Agent* and the social welfare

	Utility of Hardware Producer Agent	Percentage	Social Welfare	Percentage
Self-Interested	583	1.0	1679	1
Completely-Cooperative	395	0.68	1887	1.13
Half-Cooperative	487	0.83	1831	1.09

Table 3.4: Utility of *Transporter Agent* and the social welfare

	Utility of Transport Agent	Percentage	Social Welfare	Percentage
Self-Interested	847	1.0	1675	1.0
Completely-Cooperative	803	0.95	1846	1.10
Half-Cooperative	818	0.97	1751	1.05

be successfully finished. When one agent is *completely cooperative*, it sacrifices it own utility, but task *Produce_Computer* may still fail because the other agent is not cooperative, so the utility of *Computer Producer Agent* does not increase as expected, and the global utility does not improve. This happens when the completion of a task is spread over more than two agents. The information from *Computer Producer Agent* about its utility increase is only an estimation; it depends not only on task *Get_Hardware* for *Hardware Producer Agent*, but also relies on task *Shipping_Computer* for *Transporter Agent*. In this situation, if *Hardware Producer Agent* has no knowledge about the attitude of *Transporter Agent*, it may not be a good idea to be *completely cooperative* towards *Computer Producer Agent*. The above data also shows that the utility of *Transporter Agent* does not decrease as much as *Hardware Producer Agent* when it becomes cooperative or *half-cooperative*. This is because task *Shipping_Computer* takes less time than the task *Get_Hardware*, so it is possible for *Transporter Agent* to accept more tasks without losing too many high reward tasks from the outside.

Table 3.2 shows some statistical results about the difference between the social welfare under different cooperative situations using a t-test. For example, the first line in Table 3.2 shows that with the 0.01 Alpha-level, we can reject the hypothesis H_o that the difference between the social welfare of the system when both agents are cooperative and the social welfare of the system when both agents are self-interested is equal to 330[11], compared to the hypothesis H_a that that the difference between the social welfare of the system when

[11]330 is 20% of social welfare under the SS situation(1649), and 180 is 11% of social welfare under the SS situation.

both agents are cooperative and the social welfare of the system when both agents are self-interested is greater than 330.

Table 3.3 shows the expected utilities of *Hardware Producer Agent* and the expected social welfare under the three possible situations: when *Hardware Producer Agent* is self-interested, completely-cooperative and half-cooperative. When *Hardware Producer Agent* chooses one attitude, *Transporter Agent* may adopt one of the three different attitudes. For example, when *Hardware Producer Agent* chooses to be self-interested, the global situation could be SS, SC, or SH. The utility number in the table is the expected value of the utilities under these three different situations. Table 3.4 shows similar information for *Transporter Agent*. Table 3.3 tells us that when a cooperative operation involves more than two agents and when the other agents' attitudes are unknown, being *completely cooperative* means sacrificing its own utility significantly and thus is not a good idea. However, it is a good choice for an agent to be half-cooperative, sacrificing less of its own utility for more global utility increase. This is an example where the lack of a complete global view can be partially compensated for by having an agent acting in a partially cooperative attitude rather than being fully cooperative. For the *Transporter Agent* which does not need to sacrifice too much to be *completely cooperative*, it should always choose to be *completely cooperative*.

3.6 Related Work

Glass and Grosz [21] developed a measure of social consciousness called "brownie points" (BP). The agent earns BP each time it chooses not to default on a group task and loses BP when it does default for a better outside offer. Defaulting on a group task may cause the agent to receive group tasks with less value in the future, and thus reduces its long term utility. The agent counts BP as part of it overall utility besides the monetary utility. A parameter $BPweight$ can be adjusted to create agents with varying levels of social consciousness. This relates to our utility mapping function associated with the *relational* MQ which can be adjusted to reflect the agent's different attitudes in negotiation. However, the *relational* MQ is agent-oriented and issue specific, so the agent can model different attitudes towards each agent and negotiation issue. Additionally, the mapping function can be a nonlinear function and describe a more complicated attitude. Their work assumes there is a central mechanism controlling the assignment of group tasks according to agent's rank (agent's previous default behavior), which is not always appropriate for an open agent environment. Instead, in our assumption, agents are all independent and there is no central control in the society.

Axelrod's work [5] has shown that stable cooperative behavior can arise when self-interesting agents adopt a reciprocating attitude toward each other. The agent cooperates with another agent who has cooperated with it in previous interactions. The idea of the reciprocity is related to our work if the *relational* MQ is used in bi-direction between agents; agent A collects some *relational* MQ from agent B and in the future the accumulated *relational* MQ could be used to ask agent B do some work for it. In this way, the *relational* MQ actually works as a quantitative measure of reciprocity. Sen developed a probabilistic reciprocity mechanism [56] in which the agent K chooses to help agent J with

certain probability p and p is calculated based on the extra cost of this cooperation behavior and how much effort it owes agent J because agent J has helped it before. There are two parameters in the formula for calculating p which can be adjusted so that the agent can choose a specific cooperation level. However, this work assumes that cooperation always leads to aggregate gains for the group, and it was based on a known cost function - that is, they know how much extra it will cost then to do X for another agent. Neither of these two assumptions is necessary in our work. Also our work deals with more complex and realistic domains where tasks carry real-time constraints and there are potentially complex interrelationship among tasks distributed over different agents.

3.7 Summary

We introduced an integrative negotiation mechanism which enables agents to interact with a spectrum of negotiation attitudes, from self-interested to completely cooperative in a uniform reasoning framework, namely the MQ framework. The agent can choose not only to be simply self-interested or cooperative, but also how cooperative it wants to be from a continuum of values. This provides the agent with the capability to dynamically adjust its negotiation attitude in a complex agent society. Experimental work shows it may not be a good idea to always be *completely cooperative* in a situation involving an unknown agent's assistance; in that case, choosing to be half-cooperative may be good for both the individual agent and also for the society. In the future, we plan to explore additional questions using this framework, such as: how should an agent choose its negotiation attitude based on learning from past experience, and how do different attitudes affect the agent's performance and the social welfare in different organizational contexts?

CHAPTER 4

MULTI-LEVELED NEGOTIATION

A multi-leveled negotiation framework is introduced in this chapter. First we will discuss why the negotiation process should be performed at different levels of abstraction (Section 4.1). Then we will present the multi-leveled negotiation framework in Section 4.2. Examples are used to explain how this framework works in Section 4.3. Different reward models are discussed in Section 4.4. Section 4.5 shows how these different reward models affect the agent's performance. Section 4.6 summarizes this chapter.

4.1 Negotiation at Different Levels

Usually negotiation is structured as a single level process: from the proposal to the final commitment, all related issues such as finishing time, achieved quality and offered price are determined in this process. However, as the assumptions in Section 1.3 point out, the agent has multiple tasks and the tasks are complicated tasks; each task may be achieved in different ways and include a sequence of activities, some of which may require external or internal resources. The agent needs to choose which tasks to perform, when to perform them and how to perform them. The successful execution of a task may involve negotiation with other agents about sub-contracts or resource requirements. Meanwhile, since the agent works in a complex organizational context, it needs to work with other agents from a variety of different organizational positions. Hence the negotiation strategy should conform to the organizational relationship. Given all the above considerations and the uncertainty of task execution, it is difficult to construct an integrated framework in which all these issues are addressed concurrently and done so in an efficient way. So we feel it is important to have a multi-leveled negotiation framework in which the negotiation process is performed at different abstraction levels. The upper level deals with the formation of high level goals and objectives for the agent, and the decision about whether or not to negotiate with other agents to achieve particular goals or bring about particular objectives. The negotiation at this upper level determines the rough scope of the commitment (i.e. the time and the quality characteristics) and the cost of the commitment. The lower level deals with feasibility and implementation operations, such as the detailed analysis of candidate tasks and actions and the formation of the detailed temporal/resource-specific commitments among agents. The negotiation at this lower level involves the refinement of the rough commitments proposed at the upper level.

Let's look at an example to make these issues concrete. Agent α is Adam's personal assistant agent. Agent α is deigned to carry out multiple tasks corresponding to Adam's multiple goals in his life. Adam is a professor of Asian culture and language and he also has

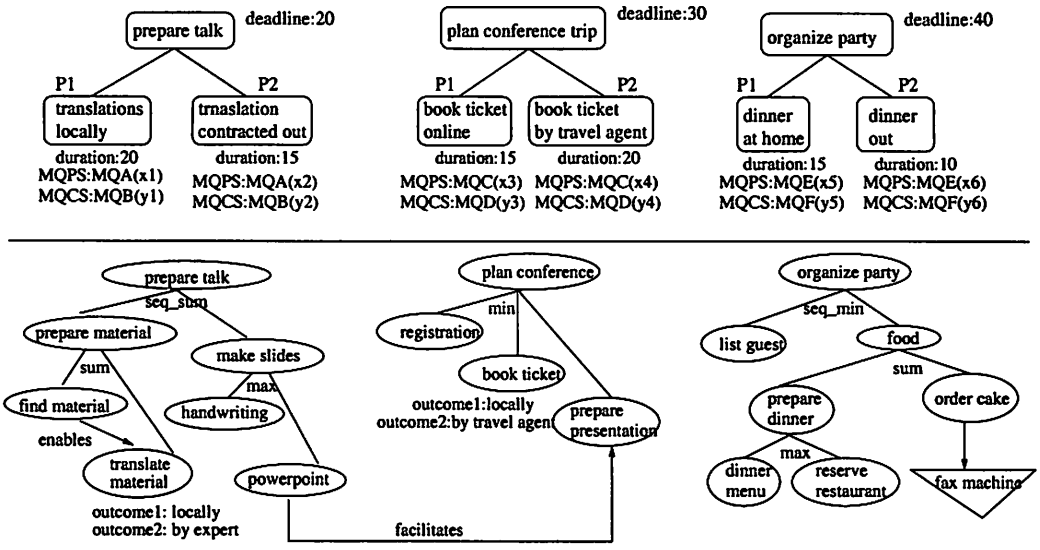


Figure 4.1: Agent A's three tasks

a family. He is asked by his department chair whether he can deliver a talk about his recent research results in the college. Also, he is planning to attend a conference in his research area. Meanwhile, his wife discusses with him the arrangement for their son's birthday party. Thus, there are three candidate tasks that appear in the agenda of agent α: prepare a talk for Adam's lecture, plan Adam's trip to a conference, and organize a birthday party for Adam's son. These tasks are associated with Adam's different roles, and contribute to different goals. The contributions of these tasks are not interchangeable. Each task has a deadline request, and also has multiple alternative ways to be performed. Figure 4.1 shows these three tasks. The higher level view describes the deadline for each task, the abstracted plans for each task, the duration of these plans and how they contribute to different goals (through the $MQPS$ and $MQCS$ set). The lower level view describes the detailed plan for each task with the specification of the execution characteristics for each primitive tasks. Figure 4.2 presents the detailed plan for task *prepare talk*.

Agent α needs make decisions about which tasks should be performed, and when and how to perform them. The possible issues that agent α can negotiate about include:

1. Negotiation with the secretary agent about when the talk should be delivered, which affects the deadline of the task *prepare talk*.

2. Negotiation with a translator agent about the task *translate material*, which includes when this task can be performed and how much it costs.

3. Negotiation with a travel agent about the task *book ticket*, which includes when this task can be performed and how much it costs.

4. Negotiation with agent B, the person assistant agent of Adam's wife, about the task *organize party*, whether agent B can perform part of this task or the whole task.

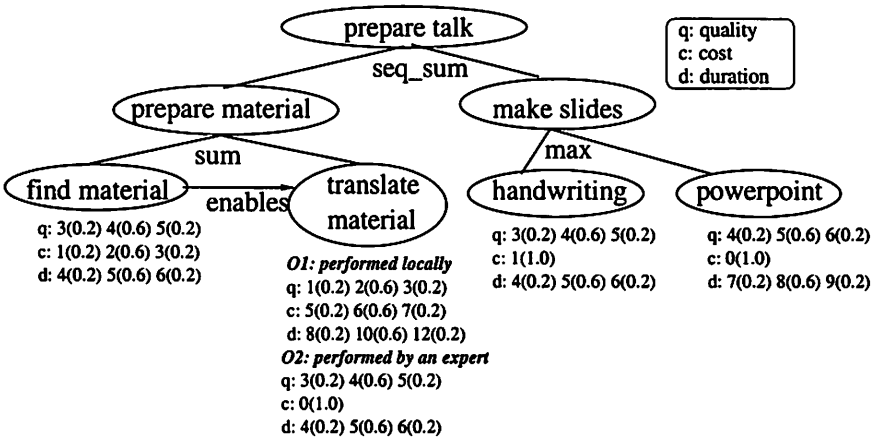


Figure 4.2: The detailed task structure of task "*prepare talk*"

These negotiation issues are inter-related, so called "multi-linked" negotiation. Also the negotiation with different agents involves different organizational relationships, hence requires appropriate negotiation strategies, as we discussed in the integrative negotiation. The previously presented mechanism for multi-linked negotiation (Chapter 2) and integrative negotiation (Chapter 3) can be applied to this problem given the following architecture and process.

It is reasonable for an agent to evaluate the importance of a commitment from the upper level. An agent has a better understanding of how a commitment could affect its local plan and hence its utility gain when it reasons about this commitment in the upper level framework. Moreover, the agent needs some initial commitments when it chooses its local plan.

In the above example, agent α needs to perform task *prepare talk*, and there are two available plans for task *prepare talk* [1]:

1. P1: prepare the talk with the translation work done locally

2. P2: prepare the talk with the translation work contracted out to a translating agent

Each plan has different quality, duration and cost characteristics. The plan $P2$ requests contracting a subtask *translate material* to another agent. From the high level view, if agent α can find another agent to perform the subtask *translate material* before time 15 and with transferred utility less than 5, the plan $P2$ is the best choice. The availability of this commitment affects agent α's local plan. If such a commitment is not available, agent α needs to choose the other plan $P1$ for task *prepare talk*. The other plan $P1$ takes a longer time for execution and hence makes it impossible for another task *plan conference trip*, which agent α has planned to accomplish, to meet its deadline. Agent α has to change its current local plan. By comparing its original local plan with the commitment on *translate material* and the other plan without the commitment on *translate material*, agent α can find out how important it is to obtain a commitment on *translate material*.

[1] To make the example simple, only two plans for each task are shown in Figure 4.1, although there are other plans for this task besides these two plans.

On the other hand, not all issues can be modeled or totally decided on the upper level. The upper level deals with the agent's high level activity plan; it lacks detailed information about each activity. Hence it is difficult to reason about the agent's detailed activities. There are two kinds of issues related to the decision-making process in the negotiation. Those issues which have strong influence on local plan selection and involve utility transferred between agents (i.e. an important non-local task or an important resource that needs to be obtained from another agent) should be negotiated first at the upper level and rough commitments should be constructed for them. However, those issues which have less influence on local plan selection and involve reasoning about the detailed structure of the low level activities, can not be modeled on the upper level and do not need to be decided on the upper level. These issues include:

1. *Internal relationships between subtasks that belong to different high level tasks.* For instance, the subtask *powerpoint* (make slides using powerpoint) that belongs to *prepare talk* facilitates the subtask *prepare presentation* that belongs to *plan conference trip* because part of the slides for the lecture can be reused in the conference presentation if the slides are done in powerpoint format. This relationship is not visible from the high level tasks. Besides, whether the subtask *powerpoint* is included in the plan for task *prepare talk* depends on which plan is selected for this task at the higher level reasoning process. However, the agent can exploit it to optimize its local plan after the high level plan is decided.

2. *Uncertainty of the execution characteristics that are not visible on the higher level.* The agent is uncertain about the task's duration, cost and quality produced when it makes a plan about the task. Expected values (or other abstraction model, such as a range with certain confidence level, see Section 4.4.2) are used in the high level planning and uncertainties are not taken into account. This leads to more efficient processing at the higher level. However, in certain situations detailed reasoning about uncertainty becomes important in making a commitment. The lower level has detailed information about the uncertainty, and since more context knowledge is available along with the process, the high level commitment can be adjusted to accommodate for uncertainty. For example, the higher level plan $P2$ for task *prepare talk* has an estimated duration of 15, which is based on the expected value of the primitive tasks' durations. Figure 4.2 shows the uncertainty information for each primitive tasks.

3. *Internal resource requirement associated with low level tasks.* For example, agent α needs to use the fax machine for task *order cake* (Figure 4.1), but it shares the fax machine with several other agents. Given the knowledge of the general usage of fax machine, the agent knows it is unnecessary to reserve the fax machine when it builds its high level plan. But when the agent comes to arrange its local activities, it should taken this resource constraint into consideration.

Considering the above issues, the agent may need to revise its higher level commitments through the lower level negotiation and additionally to reorder its lower level activities, so

as to optimize it local plan and commitments, reduce failure possibilities, avoid conflicts and achieve higher utilities.

To summarize, the basic idea of multi-leveled negotiation is the following. First agents negotiate on the upper level to build rough commitments, which are the foundation for other planning and coordination/negotiation activities. As a result of these activities, more constraints are added and more detailed information is available from the lower level; these rough commitments then are refined as a result of this new information.

In section 4.2, we will discuss the multi-leveled negotiation framework in detail.

4.2 Multi-Leveled Negotiation Framework

The multi-leveled negotiation is performed at different abstraction levels. In this thesis work, the MQ framework [67] is used for the higher level representation, while the TÆMS framework [16] is used to support the lower level reasoning process. However, the basic approach is not restricted to these two frameworks, and we feel they can also be applied to other suitable architectures.

4.2.1 Supportive Frameworks

First we would like to briefly review the major features of these two frameworks. In the MQ framework, the execution of a task contributes, in a quantitative manner, to the achievement of one or more agent's objectives. As part of this framework, there is a way of mapping this contribution to an overall utility increase associated with the potential execution of a task, given the agent's current state of achievement of different objectives. This enables the agent to compare tasks that are associated with different organizational goals, or tasks motivated by self-interested reasons to cooperative reasons. Each agent has a set of MQs or motivational quantities that it tracks and accumulates. MQs represent progress toward organizational goals and in certain cases may be used as a medium of exchange. MQs are produced and consumed by task performance where the consumption or production properties are dependent on the context. For each MQ_i belonging to an agent, it has a preference function or utility curve, U_{f_i}, that describes its preference for a particular quantity of the MQ. Different agents may have different preferences and organizational goals or directives.

MQ Tasks are abstractions of a partial order set of primitive actions that the agent may carry out. MQ tasks may have *deadlines* and *earliest start times*. Each MQ task consists of one or more MQ alternatives, where each alternative corresponds to a different performance profile of the task. Each alternative requires some time or *duration* to execute, produces some quantity of one or more MQs, called the MQ *production set* ($MQPS$), and consumes some quantity of MQs, called the MQ *consumption set* ($MQCS$).

The TÆMS task modeling language [16] (See Figure 4.4) is a domain-independent task modeling language. The agent's candidate tasks are described in hierarchical structures with alternative ways of accomplishing tasks. The primitive tasks (methods) are characterized by three features: quality, duration and cost via discrete probability distributions. Quality describes the contribution of a particular method to overall problem solving. It is

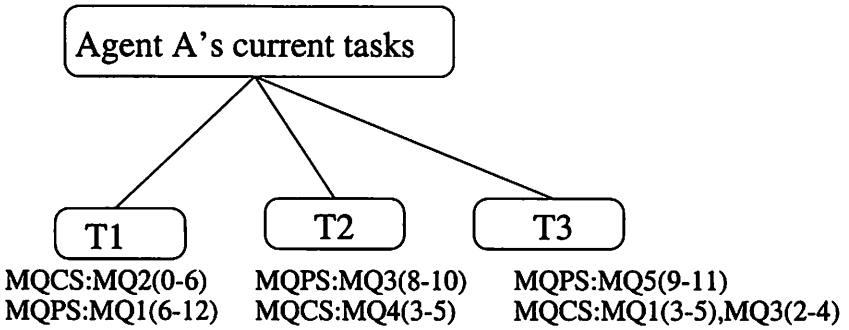


Figure 4.3: An example of MQ tasks

a domain-independent concept. Different applications have different notions of what the concept of quality models. Duration describes the amount of time that the method will take to execute, and cost describes the financial or opportunity cost associated with the performing of this action. The qaf (quality accumulation function) associated with each task describes how the qualities of its sub-tasks contribute to the quality of this task.

Hard and soft interactions between tasks, called NLEs (non-local effects), are also represented in TÆMS and reasoned about during scheduling and negotiation. Hard task interactions delineate hard precedence constraints such as *enables* and *disables*. Soft task interactions denote situations where the result of one activity can *facilitate* or *hinder* another activity. Task resource consumption and production behaviors are modeled in TÆMS via *consumes* and *produces* task/resource NLEs - these NLEs describe the quantity of resources consumed or produced by task execution. Resource requirements of methods are also explicitly modeled in TÆMS framework.

4.2.2 Overview of Basic Ideas

The MQ model [67] describes the agent's organization knowledge about task utility but it does not support detailed model of tasks and their interactions, and lacks of representation of the uncertainty characteristics and resource requirements of tasks, which belong to the TÆMS [16] model. The proper integration of these technologies enables agents not only to reason about the organizational concerns but also to handle detailed feasibility analysis and implementation of tasks.

An agent has its MQ level view of its local activities, which is a set of potential MQ tasks, each associated with certain $MQPS$ (the type and amount of MQ this task produces) and $MQCS$ (the type and amount of MQ this task consumes), which can be mapped into the agent's utility given the agent's current MQ state. For example, Figure 4.3 shows that agent A has three MQ tasks, $T1$, $T2$ and $T3$. $T1$ produces $MQ1$ from 6 units to 12 units, and it consumes $MQ2$ from 0 units to 6 units. The amount of the MQ varies depending on what plan is used to accomplish task $T1$. For each MQ task T, there is a TÆMS task structure that describes the detailed activities for this task, i.e. the task structure $TG1$ in Figure 4.4 describes the detailed activities in task $T1$. Different plans to accomplish the MQ task T can be generated from the TÆMS task group TG by the DTC scheduler, and

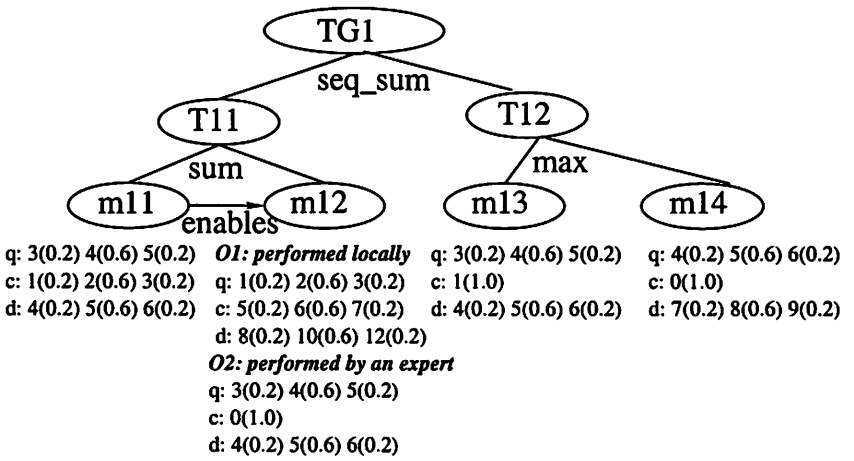


Figure 4.4: Task structure for T1

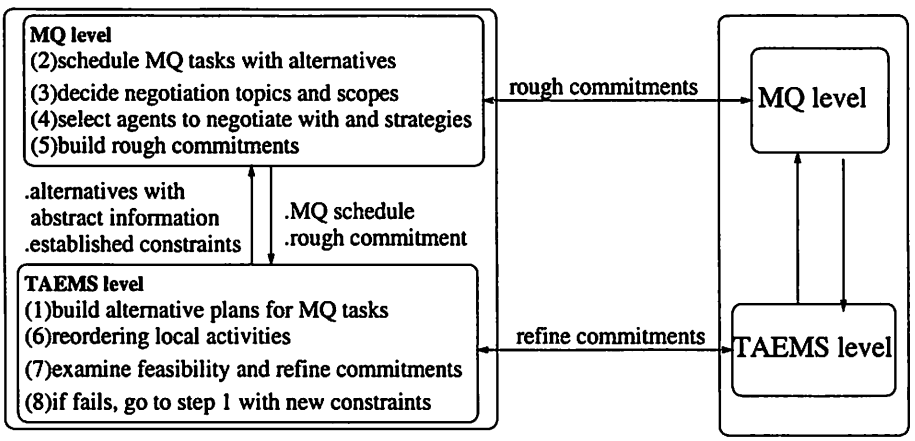


Figure 4.5: two-level negotiation framework

each plan has different quality, duration and cost characteristics that affect the $MQPS$ and $MQCS$ of the task T (See Figure 4.6). This is the first step [step 1] shown in Figure 4.5, which describes the two-level negotiation framework.

The extended MQ scheduler generates a partial order schedule that indicates what tasks the agent should attempt to execute, what plans are used to execute these tasks, and the execution ordering. This schedule represents the agent's best choice about what activities it should do to maximize its local utility increase [step 2]. Based on these schedules, the agent can reason about the utility of a specific commitment (i.e. contracting a task out to another agent, performing a task for another agent, or receiving external resources needed by one of its tasks). Negotiation on the MQ level is a multi-dimensional negotiation that includes the amount of the transferred MQ, the temporal constraints of the commitment and the quality constraints of the commitment [step3]. Also, the agent can select which agents to negotiate with and the appropriate negotiation strategy according to organizational relationships and the negotiation issues [step 4]. A partial order schedule makes it possible for the agent to reason about how a commitment affects the flexibility to modify the execution constraints on other local activities and the relationships among multiple related negotiation issues. The agent manages the multi-linked negotiation in this level using the technology presented in Chapter 2. The integrative negotiation mechanism described in Chapter 3 also fits into this step by introducing and reasoning about the *relational* MQ. The MQ level negotiation builds rough (partial-specified) commitments for those issues that should or could be reasoned about the MQ level [step 5].

After building a local MQ schedule and rough commitments on the MQ level, the agent reorders its local activities on the TÆMS level [step 6]. Low level relationships among TÆMS tasks/methods and detailed resource constraints are taken into account in this reordering process. In this reordering process, the agent could optimize its local schedule by taking advantage of the interrelationships among low-level tasks/methods. Also the agent can verify the feasibility of its local schedule given rough commitments from the MQ level and those additional constraints from the TÆMS level [step 7]. A partial order schedule is also used to manage and reason about these relationships and constraints on the TÆMS level. Negotiation on the TÆMS level involves refining those rough commitments as needed when:

1. There are conflicts or potential conflicts among commitments and local activities caused by additional constraints (such as a local resource constraint) or uncertainties in real-time execution.

2. It is possible to reduce local cost or increase local utility by refining a commitment.

If the agent can find a feasible local schedule by reordering and renegotiation on the TÆMS level, it can execute its local schedule and perform all of its commitments. If unexpected events cause conflict in the execution process, the agent needs to check if the conflict can be solved by refining any commitments. Otherwise, if the conflict can't be resolved given all current constraints, the agent needs to discard some commitments (decommits), establish other commitments on already scheduled local activities and go back to the MQ level to reschedule, and possibly result in constructing new commitments [step 8].

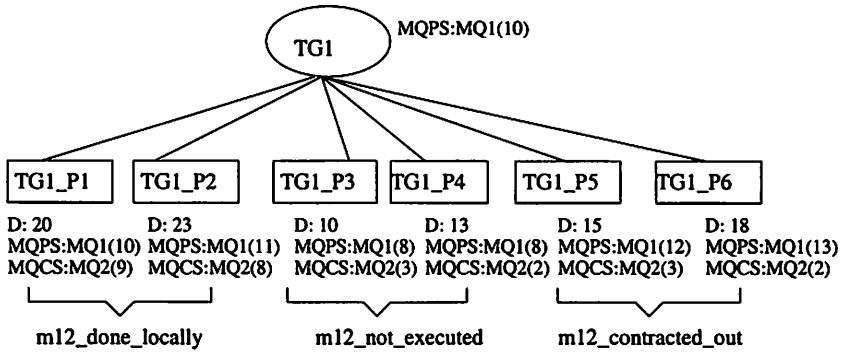


Figure 4.6: Task $T1$'s alternatives

We will discuss this two-level negotiation in more detail using examples in Sections 4.3.

4.3 Through the Process

4.3.1 DTC Scheduler Builds Alternatives

The Design-To-Criteria (DTC) scheduler [65] is a domain-independent scheduler that aims to find a feasible schedule that matches the agent's particular criteria request. In this research, it will be used off-line to build a library of alternative plans for achievement of a TÆMS task group. For example, agent A has three MQ level tasks $T1$, $T2$ and $T3$, which are mapped into the task groups $TG1$, $TG2$ and $TG3$ in the TÆMS model. There is a subtask $m12$ of $TG1$ (See Figure 4.4) that potentially can be contracted to another agent who is an expert on task $m12$.

The DTC scheduler works on $TG1$ according to the following different assumptions:

1. $m12$ is executed locally;
2. $m12$ is not executed;
3. $m12$ is contracted to another agent.

These assumptions can be combined with different quality, cost, and duration scheduling criteria to generate the following set of alternative plans:

- $TG1\_P1 : (m11, m12, m13), quality = 10, cost = 9, duration = 20$;
- $TG1\_P2 : (m11, m12, m14), quality = 11, cost = 8, duration = 23$;
- $TG1\_P3 : (m11, m13), quality = 8, cost = 3, duration = 10$;
- $TG1\_P4 : (m11, m14), quality = 8, cost = 2, duration = 13$;

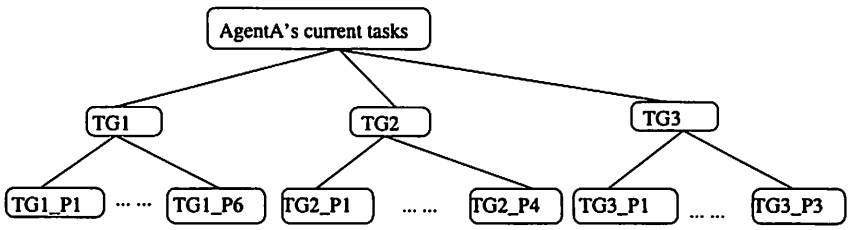


Figure 4.7: MQ level tasks

- $TG1\_P5 : (m11, [m12]^2, m13), quality = 12, cost = 3, duration = 15;$
- $TG1\_P6 : (m11, [m12], m14), quality = 13, cost = 2, duration = 18.$

Each plan has different q, c, d characteristics, corresponding to an MQ level alternative with different duration, $MQPS$, and $MQCS$, as shown in Figure 4.6. In this example, the following functions describe how the quality and cost characteristics of a plan P_n are mapped into the $MQPS$ and $MQCS$, for task $T1$:

$$MQPS : MQ1(P_n) = quality(P_n)$$
$$MQCS : MQ2(P_n) = cost(P_n)$$

This is a simple example of the mapping function. However, the mapping function could be more complex using more features such as: the likelihood to meet the deadline, the maximum derived quality rather than the expected, etc. The structure of the function also depends on the problem-solving context.

For those plans that need to contract $m12$ to another agent, such as $TG1\_P5$ and $TG1\_P6$, the MQCS does not include the cost for contracting the task $m12$, because the cost is unknown at this time. Similarly, different plans are generated for task $T2$ and $T3$.

This abstraction process can be done off-line, and these alternative plans can be stored in the agent's database. Not all alternatives are used in the MQ level scheduling process. A set of plans are selected according to the current problem-solving context. For example, if the current minimum quality request for the task is 10, then those plans with achieved quality less than 10 are discarded and not used by the MQ scheduler.

4.3.2 MQ Level Scheduling

The MQ level scheduler does scheduling for these alternatives of $T1$, $T2$ and $T3$ to find the best schedule MQ_S1 that provides the agent the most utility increase from its current state (Figure 4.7). If the plan $TG1\_P5$ or $TG1\_P6$ ($m12$ is contracted out) appears in the scheduler MQ_S1, agent A needs to consider contracting $m12$ to another agent; otherwise, agent A may choose to execute $m12$ locally or not to perform $m12$ as the schedule MQ_S1 recommends. Suppose the best schedule MQ_S1 includes the $TG1\_P5$ plan:

[2] $[m12]$ in the plan denotes that $m12$ is performed non-locally

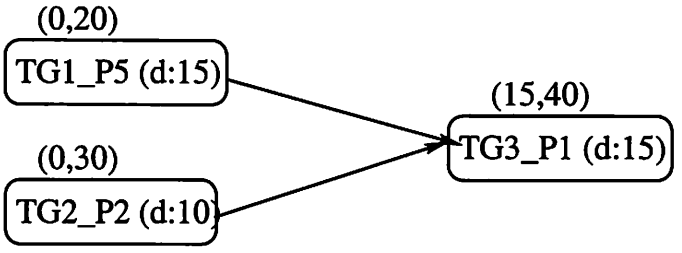


Figure 4.8: MQ level partial order schedule

$TG1\_P5$[duration:15[3] earliest start time:0 deadline:20]
$TG2\_P2$[duration:10 earliest start time:0 deadline:30]
$TG3\_P1$[duration:15 earliest start time:10 deadline:40]

This is a partial order schedule (See Figure 4.8). $TG1\_P5$ and $TG2\_P2$ need to be finished before $TG3\_P1$ starts. The reason is that $TG3\_P1$ consumes the MQs produced by $TG1\_P5$ and $TG2\_P2$. This partial order schedule can be expressed graphically as shown in Figure 4.8.

Agent A compares the utility of the best schedule including the contracting plan of $m12$ (MQ_S1) with the utility of the best schedule without the contracting plan of $m12$ (MQ_S2):

$TG1\_P4$[duration:13 earliest start time:0 deadline:20]
$TG2\_P2$[duration:10 earliest start time:0 deadline:30]
$TG3\_P1$[duration:15 earliest start time:10 deadline:40]

The difference is the utility gained by contracting $m12$ to another agent.

Marginal_Utility_Gain($m12$) = Utility(MQ_S1) - Utility(MQ_S2)

Marginal utility gain specifies the local utility increment by contracting this task to another agent. On the other hand, *marginal utility cost* specifies the local utility decrement for the contractor agent by performing this task without considering the potential benefits the contractor agent can get from the transferred MQ with the task. In other words, marginal utility cost measures the cost for the contractor agent to perform this task, in terms of resource cost, financial cost and opportunistic cost. More formal definitions of these two concepts can be found in Section 5.3.1.

The basic constraint of the quality request and the temporal constraint of $m12$ is established based on the TÆMS level schedule ($TG1\_P5$) and the MQ schedule (MQ_S1). Suppose in the $TG1\_P5$ schedule, the quality request of $m12$ is 10, and the abstraction of the schedule $TG1\_P5$ is (5, $m12$, 5); it means there are some activities of duration 5 that need to be done before $m12$ and some activities of duration 5 that need to be done after $m12$. The above information comes from the pre-analysis of the plan $TG1\_P5$. Combined

[3]The duration of a plan describes how long it takes to execute this plan. It includes both the local process time and the non-local process time. For plan $TG1\_P5$, there are 5 time units for $m12$ which is non-local process time because $m12$ is executed by another agent. It is possible for agent A to arrange other local tasks in this period of time. However, the MQ scheduler does not reason at this level of detail. The partial order scheduler which works at the TÆMS level, checks all local activities and makes sure the local process time is not wasted.

with the temporal constraint of $TG1$-P5 in the schedule MQ_S1 [0, 20], the temporal scope of $m12$ is [5, 15]. It leaves 5 units time before $m12$ and 5 units time after $m12$. These constraints are very preliminary; if there are other constraints added to other activities, the scope may need to be refined based on the TÆMS level rescheduling (see example in Section 4.3.4). The quality request is only an estimation because the agent does not know what quality the other agent may achieve for this task.

Agent A posts this task allocation proposal as:
$m12, quality - request : 10, time - scope : [5, 15]$

4.3.3 MQ Level Negotiation

The negotiation on the MQ level includes multiple issues. The first issue is the amount of the transferred MQ when the non-local task NL[4] is performed by the contractor agent at the request of the contractee agent. Another issue is the plan selected for performing task NL, including the start time, the completion time and the achieved quality of NL. Also, the agent needs to select an appropriate reward model that takes into account the possible further refinement of the rough commitment.

4.3.3.1 Transferred MQ

There are three different models for the MQ transferred with the allocated task:

1. *Fixed MQ.* The type and the amount of the transferred MQ is fixed. It is determined by the organizational relationship between the contractor agent and the contractee agent. There is no need to negotiate about the transferred MQ, but the contractor agent and the contractee agent may negotiate about the possible approach (the specified quality request, start time and finish time) of the task NL. This model implicitly represents organizational authority relationships among agents by defining how the contractor agent values the task requested by the contractee agent. Obviously, the more MQ associated with the task, the more authority the contractee agent has over the contractor agent.

2. *Negotiated MQ.* The approach to perform task NL is fixed in terms of the desired start time, finish time and the quality to be achieved. The contractor agent and the contractee agent negotiate about the type and amount of the transferred MQs depending on the contractor agent's current available MQ and the contractee agent's preference.

3. *Dynamic MQ.* In this type of negotiation, the agents negotiate about both the approach and the transferred MQ of NL. For each different approach to accom-

[4] NL represents a task that needs to be performed by another agent. In the above example, NL task actually is task $m12$.

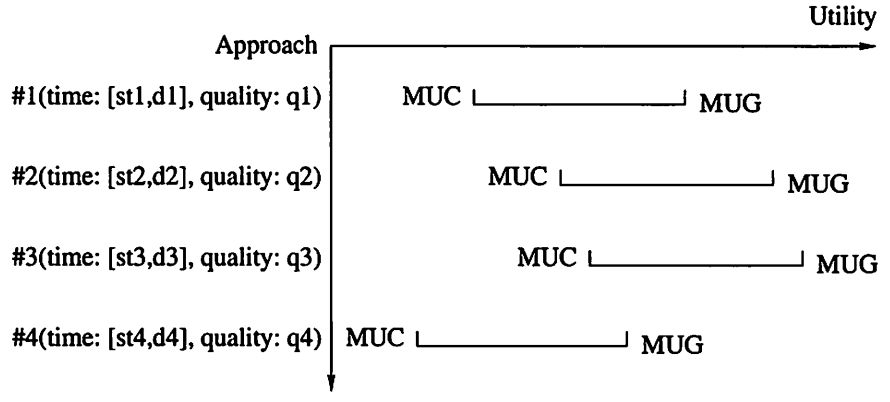


Figure 4.9: MQ level negotiation

plish task NL, the *m*arginal_utility_gain[5] ($MUG(NL)$) and the *m*arginal_utility_cost[6] ($MUC(NL)$) are different, so the MQ value space for negotiation ($[MUC, MUG]$) is different too. Also for a certain approach, the agents may negotiate about the type and the amount of transferred MQ in the corresponding MQ value space as in the model of *Negotiated* MQ.

Figure 4.9 shows the two possible negotiation dimensions in the MQ level. In the model of *Fixed* MQ, the transferred MQ is fixed: the negotiation is only about the approach; in the model of *Negotiated* MQ, the approach is fixed: the agents are searching for an agreement point in the $[MUC, MUG]$ scope by negotiating about the type and amount of transferred MQ; in the model of *Dynamic* MQ, the negotiation is performed on both dimensions.

4.3.3.2 Different Approaches

A certain approach specifies the lowest achieved quality, the earliest start time and the finishing time for the task. All these issues can be varied in this negotiation process to construct different approaches.

The contractor agent A builds the first proposal [NL, quality: q1, earliest start time:s1, deadline: d1] based on the best MQ level schedule (MQ_S1) and the selected plan ($TG1\_P5$). This proposal is evaluated by the contractee agent B. If this proposal is rejected by agent B, it will return a counter-proposal [NL, quality:q2, earliest start time: s2, deadline: d2]. Agent A revises its local plan MQ_S1 based on this counter-proposal and evaluates the utility of the new plan MQ_S1*. There is a quality accumulation function that maps NL's quality to $T1$'s quality. This function is constructed based on the structure of $T1$ and is

[5]Marginal_utility_gain represents the local utility increment for the contractee agent by having task NL performed with a specific approach, without considering the transferred MQ with task NL. The value of the transferred MQ can't exceed the marginal_utility_gain.

[6]The local utility decrement for the contractor agent by performing task NL with a specific approach, without considering the transferred MQ with task NL. The value of the transferred MQ must go beyond the marginal_utility_cost.

available to the agent. The quality of $T1$ is then mapped to $MQPS(T1)$, that determines the utility of the new schedule MQ_S1*. If the value of the transferred MQ in the counter-proposal is less than $(Utility(MQ\_S1*) - Utility(MQ\_S2))$ (MQ_S2 is the next best schedule; it may or may not need to contract out NL), then the counter-offer is acceptable; otherwise, it is rejected.

Agent A can reason about the influence of the start time and the completion time of NL in the counter-proposal on its local plan using the partial order schedule. The influence includes: how it affects the temporal constraints of each task and how it affects the flexibilities of other activities and the local schedule.

Similarly, agent B can evaluate the flexibility of the proposal (or the counter-proposal) given the estimation of the execution time of the task. The flexibility can be calculated by comparing the $[est, dl]$ range of the duration of the task (d). As the range grows, the flexibility of the commitment grows. The following formula can be used to calculate the flexibility of a commitment ($F(c)$):

$F(c) = \frac{dl(c) - est(c) - d}{d}$

In the above example, suppose the estimated duration of NL is 5 ($d = 5$), the commitment c1 with range [5, 15] has freedom $F(c1) = 1.0$; the commitment c2 with range [10, 15] has freedom $F(c2) = 0$. When the flexibility of the commitment becomes bigger, it is easier for agent B to arrange its other local activities and achieve success on its other negotiation issues; however, it is more difficult for agent A to arrange its other activities and achieve success on its other negotiations. So the value/cost of a commitment is also related to the flexibility of the commitment. If agent A wants to keep more flexibility for itself and reduce the flexibility of the commitment, it needs to pay more to agent B. If agent A does not need too much local flexibility because it has a lot of certainty about the durations of its other local activities, it can give more flexibility to agent B which makes agent B's life much easier; in this case agent A can pay less to agent B for this less flexible (more flexible for agent B) commitment. Both agents can reason from their current states to decide if they need more flexibility or if they need a cheap commitment.

4.3.3.3 Rough Commitment and Reward Models

Agents build rough commitments as a result of the MQ level negotiation. Future refinement as result of the lower level (TÆMS) negotiation is possible given the range specified by the rough commitment. The refinement will affect the flexibility of the commitment and hence affect the value/cost of the commitment. Thus agents need to negotiate over the reward model which specifies how the refinement is related to the value of the transferred MQ. Since the reward model is related to the negotiation on both levels, we will discuss this in detail in Section 4.4.1.

4.3.4 TÆMS Level Negotiation

Agent A reorders its TÆMS level tasks based on the plans chosen in the MQ level schedule. All methods not included in the MQ level schedule are eliminated from the task group and the tasks are associated with temporal constraints from the MQ level schedule.

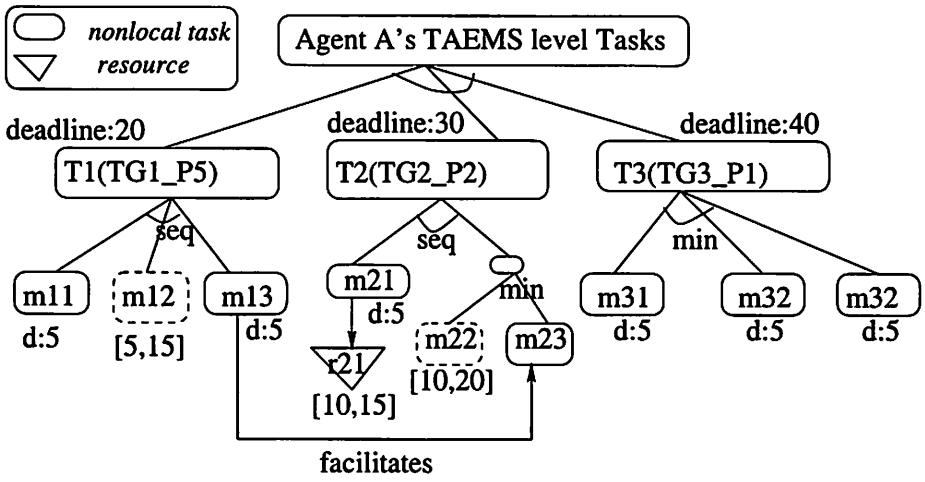


Figure 4.10: TÆMS level tasks

Figure 4.10 shows agent A's current tasks and the required negotiation issues. Agent A currently has three tasks, $T1$, $T2$ and $T3$. All methods appearing in this figure are those constructing the plan $TG1\_P5$, $TG2\_P2$ and $TG3\_P1$. $T1$ has a deadline of 20; $T2$ has a deadline of 30, and $T3$ has a deadline 40. $T1$ and $T2$ need to be finished before $T3$ starts. These constraints come from the MQ level scheduling. Also there are two commitments built at the MQ level for the non-local methods m12[5, 15] and m22[10, 20]. The agent tries to satisfy all these constraints when arranging its local activities. However, there may be other constraints that agent A needs to consider. These constraints come from the resource requirements and the relationships among those subtasks that belong to different high-level tasks: they are not visible to the MQ level scheduler so they are not reflected in the MQ level schedule. Two examples are shown in Figure 4.10:

1. There is a facilitates relationship between $m13$ and $m23$. If agent A can complete m13 before it performs $m23$, the execution of $m23$ will be facilitated in terms of getting better quality, spending shorter duration or lower cost. So agent A needs to add this additional temporal sequence constraint $[m13 \rightarrow m23]$ into its partial order schedule if it wants to exploit this facilitates relationship (shown in Figure 4.11).

2. The execution of method $m21$ needs the resource $r21$. The resource $r21$ may be managed by a resource manager or may be shared with other agents. Agent A needs to find out what time $r21$ is available so it can arrange the execution time of method $m21$.

The reordering process considers all methods contained in the MQ level schedule. It takes into account the interrelationships among tasks, the resource request constraints and the rough commitments built at the MQ level negotiation. For example, resulting from the MQ level negotiation, agent B will perform task $m12$ for agent A between time 5 and 15, and agent C will perform task $m22$ for agent A between time 10 and 20. Given that the resource $r21$ is only available from time 10 to 15, agent A can't find a feasible local

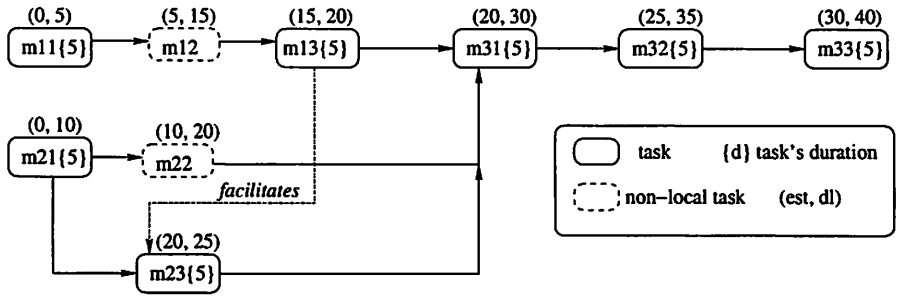


Figure 4.11: TÆMS level partial order schedule

schedule. One solution is to negotiate with agent C to push the start time of $m22$ to 15 instead of 10 (suppose the duration of $m22$ for agent C is 5). If the commitment on $m22$ between agents A and C is the *pre-paid flexibility*[7] model, then agent C would accept this request. Otherwise, if the commitment is associated with the *dynamic flexibility* [8] model, agent C needs to reason about its local partial order schedule to determine if it can grant this request. If it can, agent C will get extra MQ from agent A as they have agreed on at the MQ level negotiation. If this refinement negotiation is successful, agent A can generate a new feasible local schedule:

m11[0-5] *m12[5-15]* m13[15-20]

m21[10-15] *m22[15-20]* m23[20-25]

m31[25-30] m32[30-35] m33[35-40]

Besides the additional constraints caused by resource requirements and the relationships among those subtasks that belong to different high-level tasks, the other reason for TÆMS level negotiation is the uncertainty of task execution. More details about the uncertainty will be discussed in Section 4.4.2.

4.3.5 MQ Level Rescheduling

If the refinement negotiation fails and agent A can not find a feasible local schedule given all local constraints, agent A has the following choices:

1. Select a similar plan with a different schedule and try to solve the conflict. For example, if the current plan for task $T2$ is TG_P2, agent A checks if there is another plan for task $T2$ with similar q, c, d characteristics but does not require the resource $r21$. If such a plan can be found, the constraint caused by the requirement of resource $r21$ can be removed.

[7]See definition in Section 4.4.1.

[8]See definition in Section 4.4.1.

2. Discard some impossible tasks/commitments. If the current schedule is already late and there is no way to prevent a task from missing its deadline, the agent can remove the rest of this task from its current schedule and leave room for other tasks.

3. Reschedule at the MQ level, given the current commitments, tasks, and newly arrived tasks. Before rescheduling at the MQ level, the agent needs to make a decision about what commitments should be preserved and what commitments should be decommitted. Some commitments are established and preserved, and the related MQ level tasks are associated with corresponding temporal constraints, while other commitments are decommitted. This decision making process should take into account the importance of the commitments, the urgency of the commitments, the decommitment penalty and the scarcity of the resource.

The first two choices cause the agent to generate a schedule which is different from the original one that was optimal given the knowledge at the time of scheduling; hence the agent's utility achievement won't be as good as it expects. However, the choice of rescheduling on the MQ level may involve much higher cost compared to the first two choices, although it promises to provide an optimal solution given all current knowledge. So the agent needs to compare the loss of utility as a result of following a sub-optimal solution to the cost of rescheduling.

4.4 Reward Models and Uncertainties

4.4.1 Reward Models

Agents build rough commitments as a result of MQ level negotiation. They are rough commitments since the specifications can be ranges rather than points; these ranges allow further refinement. For example, a rough commitment c could specify the temporal constraint for the contracted task NL to be started and completed somewhere between $[t1, t2]$. If $F(c) > 0, t2 > t1 + d$ ($F(c)$ denotes the flexibility of c; d denotes the estimated duration of NL), it is possible to refine this commitment by restricting this range to $[t1 + x, t2 - y]$, $(t2 - y - t1 - x \geq d)$; hence the flexibility of the commitment c is reduced. Because the flexibility is related to the value/cost of the commitment, the agents need to come to an agreement on how the latter refinement is related to the value of the transferred MQ. There are two possible models:

1. *Pre-paid flexibility* model. The contractee agent A pays $v1$ of MQ_i for the contractor agent B to perform task NL during any time period (not shorter than d) within $[t1, t2]$ as agent A requests. This agreement provides agent A with the freedom to further refine this commitment, and agent B agrees to accommodate any request from agent A within the pre-defined range. No matter what request agent A will make, or even if agent A does not make any further requests, agent B will receive $v1$ of MQ_i as decided in the rough commitment.

2. *Dynamic flexibility* model. The contractee agent A pays $v2$ of MQ_i for the contractor agent B to perform task NL within the range of $[t1, t2]$. If agent A requests a

restriction on this range to $[t1 + x, t2 - y]$, $(t2 - y - t1 - x \geq d)$ and if agent B could accept this request, agent A will pay $((x + y) * \beta + 1) * v2$ of MQ_i to agent B. Agent B would decide to accept this additional refinement request or not, according to its current problem-solving context. If agent B does not accept this request, it is still obliged to perform NL during $[t1, t2]$ and in turn is guaranteed to get $v2$ of MQ_i as the rough commitment defines.

These two models provide different degrees of freedom for the agents. The agents can choose a model according to the constraints and uncertainties of their local activities during the negotiation process.

4.4.2 Uncertainties

The uncertainty discussed in this section means the uncertainty in the estimation of the execution characteristics (i.e. duration, quality, and cost) of an activity. This type of uncertainty can be represented as a statistical distribution:

$V : \{v_1(p_1); v_2(p_2); ...; v_n(p_n)\}$

meaning variable V has chance of p_i of being the value of v_i $(i = 1, ...n)$.

Expected Value of variable V :

$$E(V) = \sum_i p_i v_i$$

Measure of Uncertainty of variable V :

$$MU(V) = -\sum_i p_i * log(p_i) * \frac{|v_i - E(V)|}{E(V)}$$

Probability of Above Expectation of variable V :

$$PAE(V) = \sum_{i|v_i > E(V)} p_i$$

Measure of Above Uncertainty of variable V :

$$MAU(V) = \sum_{i|v_i > E(V)} p_i * (v_i - E(V))$$

For example:

$X = \{3(0.3); 5(0.4); 7(0.3)\}$;
$E(Y) = 5$; $MU(X) = 0.125$; $PAE(X) = 0.3$; $MAU(X) = 0.6$;
$Y = \{4(0.1); 5(0.8); 6(0.1)\}$;
$E(Y) = 5$; $MU(Y) = 0.04$; $PAE(Y) = 0.1$; $MAU(Y) = 0.1$;

X and Y have the same expected value, but X has bigger uncertainty than Y .

4.4.3 Handling of the Distribution Explosion Problem

When uncertainty is taken into account in the scheduling process, there is a distribution explosion problem. For example, in Figure 4.12, task Tc is a non-local task and tasks Ta, Tb and Td are local tasks. From the above local schedule, the earliest start time of Tc is 10 and the deadline is 20. Ta, $m12$ and Tc all have the expected duration 5. If we introduce a more detailed model of duration as a statistical distribution:

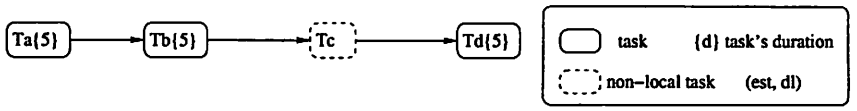


Figure 4.12: An example of one non-local task

$Ta : \{4(0.2); 5(0.6); 6(0.2)\}$
$Tb : \{3(0.1); 5(0.8); 7(0.1)\}$
$Tc : \{4(0.1); 5(0.8); 6(0.1)\}$

the expected earliest start time for task Tc will be :

$est(Tc) =$
$Ta : \{4(0.2); 5(0.6); 6(0.2)\} + Tb : \{3(0.1); 5(0.8); 7(0.1)\}$
$= \{7(0.02); 8(0.06); 9(0.18); 10(0.48); 11(0.18); 12(0.06); 13(0.02)\}$

This is a distribution of seven possible values. The distribution can contain more values when the reasoning process is longer than two steps. The following are possible approaches for handling the problem of increasing numbers of values in the distribution:

- ignore low probability events:

  $$est(Tc) = \{9(0.26); 10(0.48); 11(0.26)\}$$

- cluster the similar values using the expected value:

  $$est(Tc) = \{8.6(0.26); 10(0.48); 11.4(0.26)\}$$

- preserve the marginal values (the minimum value and the maximum value in the distribution):

  $$est(Tc) = \{7(0.02); 10(0.96); 13(0.02)\}$$

If a single number is used in the negotiation process as the earliest start time of task Tc, it can be the expected value 10 or the marginal value 13.

The expected value reflects the estimation of the end time of the previous tasks, with a greater than 50% chance of working correctly. When the duration of the previous tasks exceeds the expected value, the agents need to re-negotiate their actions. This requires a refinement of the rough commitment and a real-time coordination. It is possible for the contractor agent to attach this distribution $\{10(0.74); 11.4(0.26)\}$ to the commitment to inform the contractee agent that with a 26% chance it will need to extend the start time to 11.4. If the contractee agent accepts this commitment, it needs to prepare for this possible further refinement or coordination. On the other hand, the contractee agent would charge a greater service fee for handling this uncertainty. The marginal value reflects the latest possible time to finish the previous task. If this value is used to build the commitment, no future refinement or coordination is needed, but it makes the commitment unnecessarily over-constrained in most cases.

4.4.4 Reasoning about Uncertainty

The general idea to accommodate uncertainty in this negotiation framework is described as follows. In the low level reasoning process, uncertainties are represented as statistical distributions. The distribution explosion problem is handled using the clustering method and in the clustering process, the marginal values are preserved[9]. Uncertainty information is abstracted as the expected value, the marginal value, the *probability of above expectation* and the *measure of above uncertainty*. This abstraction information is used in the upper level reasoning process. The upper level process does not deal with the detailed distribution information. Given the marginal value and the probability of the above expectation, the agent chooses the appropriate reward model. If the *probability of the above expectation* is large (bigger than a pre-set limit) or the *measure of the above uncertainty* is large, the agent can choose the *pre_paid flexibility* model. Otherwise it can choose the *dynamic flexibility* model. The marginal value is attached to the commitment to describe that a specified item in this commitment may need to be changed by the extent of the marginal value. If the contractee agent promises to accommodate this change when requested by the contractor agent (*pre_paid flexibility* model), it can charge a higher price for this commitment but it also needs to reserve enough room in its local plan for the future change. Otherwise, the contractee agent can choose the *dynamic flexibility* model. In this way it does not promise to accommodate the future change. When the contractor agent requests a change, it checks its local plan to see if this change can be guaranteed. If so, an extra cost is added when the change really happens.

4.5 Experimental work

The experimental work in this section studies how the two-leveled negotiation mechanism affects the agent's performance compared to one-leveled negotiation. We further study how the upper level negotiation (the choice of reward model) affects the lower level negotiation and hence affects the agent's performance.

The experiments use the same supply-chain example described in Section 2.1.1. Four agents were built using the JAF agent framework [62]. New tasks were randomly generated. Uncertainties are introduced by the execution component which generates the execution time for a task according to its statistical distribution. This scenario represents a class of problems with real-time uncertainties on tasks' execution times, where some of commitments may be changed to avoid the missing of deadlines. If a task takes longer than the expected time, it may cause other tasks to miss their deadlines. The lower level negotiation occurs when this delay can be avoided by refining some rough commitments of non-local tasks. The other reasons for lower-level negotiation, such as additional constraints caused by resource requirements or reordering of the lower level tasks, didn't occur in this experimental setup; however, the two-leveled negotiation mechanism is capable of supporting re-negotiation caused by all types of reasons.

Four different policies are tested:

[9] As described in section 4.4.3.

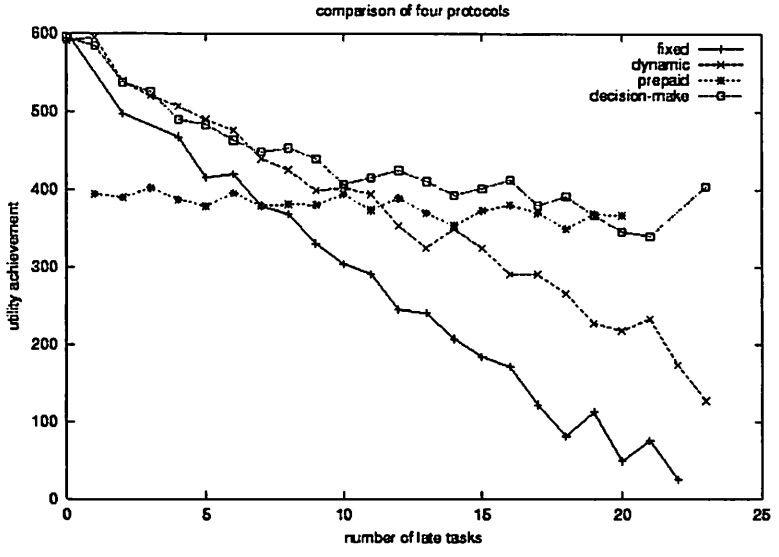


Figure 4.13: *Computer_Producer_Agent*'s **performance using different policies when uncertainty changes**

1. Fixed policy: The commitment built on the upper level (MQ level) is fixed; there is no lower level re-negotiation to refine the commitment from the upper level.

2. Dynamic flexibility policy: The agent always chooses the *dynamic flexibility* reward model in the upper level negotiation.

3. Pre-paid flexibility policy: The agent always chooses the *pre-paid flexibility* reward model in the upper level negotiation.

4. Decision-making flexibility policy: In the upper level negotiation, the agent chooses either the *dynamic flexibility* reward model or the *pre-paid flexibility* reward model according to the abstracted uncertainty information, as described in Section 4.4.4.

The entire experiment contains 225 group experiments. Each group experiment has the system running for 1000 time clicks for four times and each time the agents use one of the four different strategies. We focus on the performance of *Computer_Producer_Agent* because it is the only agent who needs to sub-contract its subtask to other agents.

Figure 4.13 shows that when uncertainty increases (the number of late tasks increases), the agent's performance decreases significantly without the lower level negotiation (using the fixed policy). The reason is that the agent can not get the expected reward without finishing the task on time; additionally it has to pay the decommitment penalty. The lower level negotiation helps the agent to adjust its previous commitment with the other agent, so as to accommodate the uncertainties and avoid missing tasks' deadlines. As the uncertainty

increases, the performance of the dynamic flexibility policy decreases, because the dynamic flexibility policy can not guarantee the success of the lower level negotiation. The other agent may accept the adjust request or not depending on its current problem solving context. With the pre-paid flexibility policy, the agent's performance is almost stable regardless of the change of the uncertainty. The agent always pre-pays for the flexibility to adjust the rough commitment whether it needs it or not. When the uncertainty is lower (the number of late tasks is less than 9), the agent actually wastes some of its potential gain by paying for flexibility it does not need. The decision-making flexibility policy brings the agent the nearly-best performance in all situations with different uncertainties, because the agent can reason about when it may need flexibility and can pre-pay for it, or when it may not need extra flexibility and can save money on the contract.

4.6 Summary

In this chapter we presented a two-leveled negotiation framework, in which the agent reasons about and negotiates over more important issues at the upper level (MQ level), and then refines the rough commitments at the lower level, in order to optimize its local plan and accommodate additional constraints and uncertainties. Examples are used to explain how a number of different technologies, such as MQ, TÆMS, and DTC can be integrated together to support sophisticated negotiation. The multi-linked, integrative negotiation technologies which are described in previous sections can also fit into this framework. Additionally, agents can choose an appropriate reward model in the higher level negotiation according to the uncertainty measure; hence the agent can pay for its local flexibility to accommodate the future uncertainty. The two-leveled negotiation framework enables the agent to handle complicated negotiation issues and uncertainties in a more efficient way.

CHAPTER 5

COOPERATIVE NEGOTIATION AS DISTRIBUTED SEARCH PROCESS

In this chapter, we will study two questions about the negotiation process. The first question is, in a multi-step negotiation process, when should the negotiation be stopped so that the cost of negotiation and the gain from the negotiation can be balanced? The second question is, how should the agent control a negotiation process with multiple attributes in negotiation? In this chapter, we focus the work on cooperative negotiation in the task allocation domain; however, the mechanisms and results are not limited to this domain.

We present a multi-dimensional, multi-step negotiation mechanism for task allocation among cooperative agents based on distributed search. This mechanism uses marginal utility gain and marginal utility cost to structure this search process, so as to find a solution that maximizes the agents' combined utility. These two utility values together with temporal constraints summarize the agents' local information and reduce the communication load. This mechanism is anytime in character: by investing more time, the agents increase the likelihood of getting a better solution. We also introduce a multiple attribute utility function into negotiations. This allows agents to negotiate over the multiple attributes of the commitment, which produces more options, making it more likely for agents to find a solution that increases the global utility. A set of protocols are constructed and the experimental result shows a phase transition phenomenon as the complexity of negotiation situation changes. A measure of negotiation complexity is developed that can be used by an agent to choose an appropriate protocol, allowing the agents to explicitly balance the gain from the negotiation and the resource usage of the negotiation.

This chapter is structured as the follows. First we will discuss what cooperative negotiation is (Section 5.1), then we will briefly introduce the TÆMS language and Design-To-Criteria (DTC) scheduler which support this work (Section 5.2). The task allocation negotiation mechanism is presented in Section 5.3. A set of protocols based on this mechanism are described in Section 5.3.2, with examples to explain how these protocols work. Experimental work is presented in Section 5.5. Some additional thoughts about the experimental results are discussed in Section 5.6. In section 5.7, we will discuss the different negotiation approaches which have been presented in this chapter and previous chapters. Section 5.8 concludes.

5.1 Cooperative Negotiation

Negotiation is a process by which two or more parties make a joint decision. The agents first verbalize demands and then move toward an agreement through a process of concession formation or search for new alternatives [37].

The negotiation research in multi-agent systems falls into two main categories, competitive negotiation and cooperative negotiation. Competitive negotiation occurs among self-interested agents [52], each trying to maximize its local utility; while in cooperative negotiation, agents try to reach the maximum global utility that takes into account the worth of all their activities. This latter form of negotiation is quite different from competitive negotiation, and can be viewed as a distributed search process. We will focus on this cooperative negotiation which, as of late, has not received very much attention in the related literature [31]. In fact, we feel there is very little work on cooperative negotiation that explicitly tries to maximize a multi-dimensional global utility function. The closest work to our knowledge is that of Moehlman et al. [36]; however their work involves a much simpler and more structured utility function that avoids quantitative reasoning about the combined utility of the agents. Additionally, their approach is not empirically evaluated in different negotiation situations.

There are different degrees of cooperation in a multi-agent system. The most extreme is "global cooperation", which occurs when an agent, while making its local decision, always tries to maximize the global utility function that takes into account the activities of all agents in the system. Global cooperation is unachievable in most realistic situations because of the number of agents and bounds on computational power and bandwidth. Thus we focus our research on "local cooperation" [30] which occurs when two or more agents, while negotiating over an issue, try to find a solution that increases the sum of their local utilities, without taking into account the rest of the agents in the system.

Furthermore, our agents negotiate over multiple attributes (dimensions) rather than over a single dimension. For example, agent A wants agent B to do task T for it by time 10, and requests the minimum quality of 8 for the task to be achieved. Agent B replies that it can do task T by time 10 but only with the quality of 6; however, if agent A can wait until time 15, it can get a quality of 12. Agent A will select the alternative it believes is better for both agents. The negotiation relates to both the completion time and achieved quality of the task, and thus the scope of the search space for the negotiation is increased, improving the agents' chance of finding a solution that increases the combined utility.

Our approach puts emphasis on a multi-step negotiation process in which agents engage in a series of proposals and counter-offers to decide whether the contractor agent will perform a task for the contractee agent by the specified time with a certain quality. This is a search for those plans and constructed schedules of an agent's local activities that increase or maximize the combined utility of the agents. We will use measures of marginal gain and marginal cost first used in the TRACONET agents [51] to structure the search. In that work, these measures were used for a single phase evaluation rather than as a basis for cooperative/distributed search among agents to find the best combined local schedules.

The cooperative negotiation process can potentially have many outcomes, depending upon the amount of effort that the agents want to expend on the negotiation. One possibility is that they will find a solution that leads to the maximum combined utility; another

possibility is that they will find a solution that increases the combined utility from their current state; while a third possibility is that they may find that either there is no solution that increases the combined utility or that they can not find one given a limited search[1].

After the negotiation starts, the agent needs to decide when to stop the process because negotiation costs accrue with time. It may stop after it gets the first acceptable solution that increases the utility or it may decide to continue looking for a better one. The agent needs to establish a balance between the negotiation cost and the negotiation benefit. There are many different possible variations of cooperative negotiation protocol, depending on the alternatives chosen above. Therefore, as part of this work we will examine these questions experimentally to produce insights about how the characteristics of the current situation affect the variant of the protocol chosen.

In the remainder of the chapter, we present our work on cooperative negotiation in the task allocation domain. First, we describe the negotiation framework, followed by the negotiation mechanism. We next discuss the experimental results obtained by using these protocols. Finally, we summarize our work and discuss future work.

5.2 Supportive Tools -TÆMS & DTC

The TÆMS language [16] is used to represent the agent's local tasks and activities (See Figure 5.4). The TÆMS task modeling language is a domain-independent framework used to model the agent's potential activities. It is a hierarchical task representation language that features the ability to express alternative ways of performing tasks, statistical characterization of methods via discrete probability distributions in three dimensions (quality, cost and duration), and the explicit representation of interactions between tasks.

The cooperative negotiation mechanism makes the assumption that a local planning and scheduling mechanism exists that can decide what method execution actions should take place and when. The local scheduler attempts to maximize a specified multi-dimensional utility function. The DTC (Design-To-Criteria) [65] scheduler is used as the agent's local scheduler in this work. It is a domain-independent scheduler that aims to find a feasible schedule that matches the agent's local criteria request. The first input for the DTC scheduler is the TÆMS task structure that describes the agent's local activities and the objective criteria used to evaluate alternative schedules. The second input is a set of existing and proposed commitments, C, that indicates that this agent will produce specific results of certain qualities by certain times. The third input is a set of non-local commitments, NLC, that are commitments made to this agent by other agents. The scheduler uses this information to find the best schedule given the objective criteria, that exploits the given non-local commit-

[1]Another possibility, which we will not consider in this work, is that the agents will recognize either at the start of negotiation or at some intermediate point that either it is highly unlikely that a solution that increases the global utility would be found or the effort to find such a solution is not worthwhile in the current context. In this case, each agent could enter a meta-level phase of the negotiation process where it could either abandon the negotiation or change the context of the negotiation by altering the set of objective criteria issues over which agents negotiate. Thus, before the negotiation, an agent could evaluate the current situation to decide if it should start the negotiation based on whether it has a good chance of increasing the global utility.

ments, honors the existing commitments and satisfies the proposed commitments as best as possible.

5.3 Task Allocation Negotiation Mechanism

In a multi-agent system, an agent may need to contract out one of its local tasks to another agent because it can't perform the task locally. This task can potentially be part of a larger activity that the agent performs in order to achieve some desired goal. The agent needs to negotiate with another agent about the appropriate time and approach to execute this task, so that the combined utility (the sum of both agent's local utilities) can be increased. By "approach", we mean a specific alternative way for another agent to perform the task which might differ in the resources (i.e. the computation time and cost) used and the quality of the solution obtained.

An agent will contract out a task to another agent if it does not have the capabilities to perform this task locally or if it is overloaded. We assume that the agent will use the TÆMS task representation of its activities to communicate with the negotiation subsystem about which task it definitely can't do locally and those tasks that it thinks may be advantageous to be performed by another agent. As part of the negotiation process, the relative merits of the option of doing the task locally or not doing it at all versus the option of contracting will be taken into account.

5.3.1 Definitions

- Contractee Agent (contractee): the agent which has a task (non-local task NL) that needs to be assigned to another agent. The contractee gains quality from this task when it is completed (TCE is the contractee's local task structure).
- Contractor Agent (contractor): the agent which performs this task for the contractee. It devotes processing time and other resources to this task without directly gaining quality (TCR is the contractor's local task structure).
- Marginal Utility Gain [NL, C] (MUG): the local utility increment for the contractee by having task NL performed with duration and quality specified as in commitment C.
- Marginal Utility Cost [NL, C] (MUC): the local utility decrement for the contractor by performing task NL with duration and quality specified as in commitment[2].

[2]To compute the marginal utility cost in a real time system, not only the actual usage of resource should be considered, but also the opportunity cost involved: when the contractor agent makes a commitment to the task NL, it loses the opportunity to perform another incoming task with higher utility, and thus the marginal utility cost may be higher than the immediate utility decrease from performing this task. Similarly when the contractee agent contracts the task NL out, it leaves itself more freedom to accept another higher utility task, hence the marginal utility gain may be higher than the immediate utility increment from the task NL.

5.3.2 Mechanism

Figure 5.1 depicts a Finite State Machine (FSM) model that describes the agents' protocol which implements the task allocation mechanism. The upper part shows the contractee's FSM; the lower part shows the contractor's FSM. The contractee agent starts the negotiation by building a proposal (Action A:*buildProposal*) and sending this proposal (Action B:*sndMsgProposal*) to the contractor agent. After receiving this proposal (*rcvMsgProposal*), the contractor agent evaluates it (Action J:*evalProposal*): if the marginal utility gain is greater than the marginal utility cost, it accepts this proposal (Action M:*sndMsgAccept*); otherwise, this proposal is rejected, the contractor agent builds a counter-proposal (Action K:*buildCounterProposal*) and sends it to the contractee agent (Action L:*sndMsgCounterProposal*). When the contractee agent gets this counter-proposal (*rcvMsgCounterProtocol*), it evaluates this counter-proposal (Action C:*evalCounterProposal*). If the counter-proposal is acceptable and there already are a sufficient number of solutions (a solution is an acceptable commitment with MUG greater than MUC), the negotiation is terminated and the contractee agent informs the contractor agent which commitment is finally built (Action F:*sndMsgFinish*); otherwise, the contractee agent generates a new proposal based on its previous proposal and the current proposal (Action D:*generateNewProposal*), and starts another round of communication.

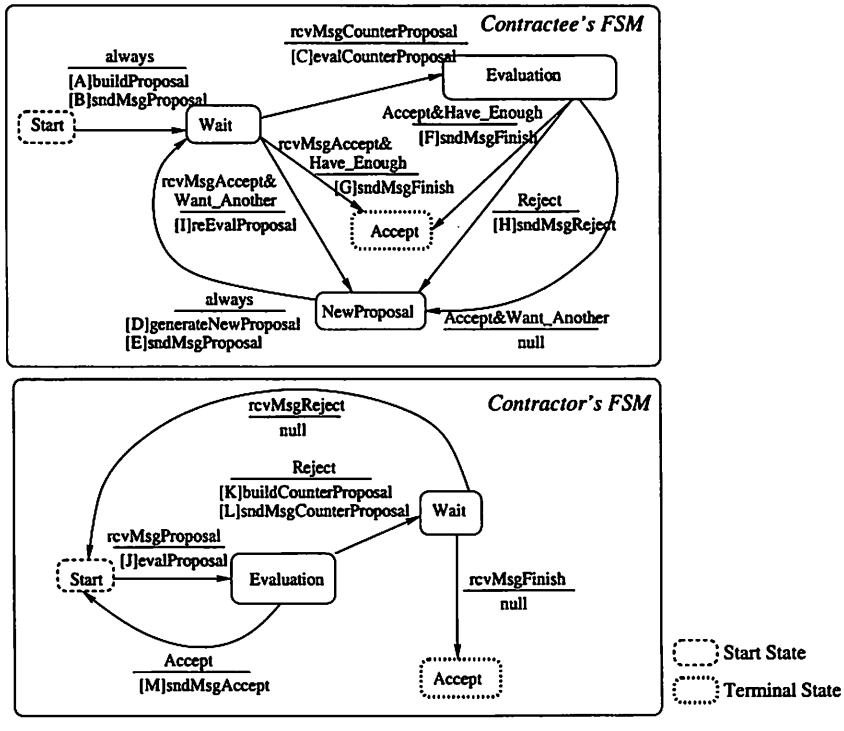


Figure 5.1: Cooperative task allocation protocol

This mechanism is actually a distributed search process: both agents are trying to find a solution that maximizes the combined utility (that is actually to maximize the marginal utility gain minus the marginal utility cost). It is not realistic to guarantee an optimal

solution given limited computational resources and incomplete knowledge (one agent does not know the other agents' situation), so the goal is to find an acceptable solution, and try to get better ones if more time is available. The contractee agent first builds an initial proposal including the time request and the quality request for the non-local task. The time request is a time range defined by the earliest possible time the non-local task can start and latest reasonable time the non-local task NL can be finished. Since there are sequence requirements and interrelationships among tasks, there are some tasks that must be finished before the non-local task can start, and there are some other tasks that can't start before the non-local task is finished. For the non-local task, the earliest possible start time is the earliest possible finish time for those tasks (pretasks) that have to be finished before the non-local task can start, the latest reasonable finish time is the latest start time for those tasks (without violating their deadline) that have to be performed after the non-local task is finished. The contractee agent gets maximum marginal utility gain during this time range, and the gain is indifferent to when the non-local task is actually executed during this range. The marginal utility gain decreases outside of this range, but it is still worthwhile to search outside of this range because the marginal utility cost for the other agent may also decrease outside of this range. So each subsequent proposal from the contractee is built from its own previous proposal by moving the time request later. The mechanism also allows for the possibility of varying NL's quality throughout the range specified by alternative ways for the contractor to accomplish the task. In this way, through additional search on these alternative time ranges, the negotiation process has an anytime character where additional time increases the likelihood of getting a better solution.

Let us describe the mechanism in greater detail. This protocol uses three functions. One generates an initial proposal by the contractee, the second generates a counter-proposal by the contractor, and the third has the contractee generate a new proposal in response to the counter-proposal. When the contractee obtains its task structure and finds that there is a non-local task NL which needs to be assigned to another agent, it builds a proposal commitment PC based on its local schedule. This commitment specifies the earliest start time, the latest finish time and the quality request for NL's execution (buildProposal function, see section 5.3.3). In addition to this information, the marginal utility gain of this commitment is also provided by the contractee. This commitment and associated information are sent to the contractor. The contractor evaluates this commitment in the context of its existing set of potential activities and other commitments as specified in its local task structure by asking a "what-if" question to the scheduler. If this commitment can be satisfied with the marginal utility gain greater than the marginal utility cost, the contractor accepts this commitment; otherwise, the contractor tries to refine this commitment (CounterProposalGeneration function, see section 5.3.3), and sends a counter-proposal CC to the contractee. When the contractee receives this counter-proposal CC, it evaluates CC by adding it to its local task structure and seeing what the resulting local schedule is. If there is a local schedule whose marginal utility gain exceeds the marginal utility cost of the counter-proposal, the counter-proposal is accepted; otherwise, it is rejected. If the counter-proposal is rejected or the contractee wants to find a better commitment, the contractee tries to improve the commitment (NewProposalGeneration function, see section 5.3.3). The improvement is a two-dimensional search process based on the time and quality requirement suggested in the previous commitment and counter-proposal from the contractor. The new commitment

is sent to the contractor and another negotiation cycle starts. As the negotiation progresses, the contractee keeps track of the number of accepted commitments and stores the accepted commitment with the highest global utility. The negotiation process ends either when the number of negotiation cycles exceeds a predefined limit or the contractee has registered that the desirable number of improvements over the original accepted commitment has been made. If the contractee has registered an accepted commitment by the time any of these events occurred, the contractee notifies the contractor of the commitment that has been finally agreed upon.

The mechanism described above also can be applied to multiple potential contractor agents. The contractee agent can start multiple parallel negotiation processes with each of the potential contractor agents, and pick the best acceptable commitment in the end.

5.3.3 Elaboration of Protocol Functions

The **buildProposal** function is used by the contractee agent to build an initial proposal PC. When the contractee finds out that there is a non-local task NL that needs to be assigned to another agent, it first performs a local scheduling process, which assumes the non-local task can be executed by the contractor agent at any time. As a result the contractee gets its local best schedule with the highest local utility achieved. It analyzes this schedule and finds the earliest start time and the latest finish time for the non-local task required by the tasks related to this non-local task. The earliest start time and the latest finish time define a range that maximizes the marginal quality gain. The length of this range is dependent on the relationships between the non-local task and other tasks, as well as the time constraints on other tasks. Besides this time range, this initial proposal also specifies the quality request for NL's execution. The contractee agent doesn't know exactly what different kinds of quality may be achieved and how long it takes or how much it costs to achieve a certain quality. The contractee only knows the range of values that task NL's quality can take, and the estimated duration of the NL. The decision about what quality to choose is important because if the initial quality request is too high, the contractor agent may fail to achieve it given the time range constraint, or even if it is achievable, the marginal utility cost may be higher than the gain; hence the proposal fails. On the other hand, if the quality request is too low, it may miss a better solution at this time. A heuristic is used to assign the initial quality request value: if the time range is much longer than the estimated duration of NL (i.e. the time range is larger than one and a half times of the estimated duration), then the quality request is set to a value higher than the average quality value (i.e. 1.2 times the average quality value); if the time range is very short compared to the estimated duration, then the quality request is set to a value lower than the average quality value; otherwise, the quality request is set as the average quality range. So the contractee agent requests a higher quality achievement if it is more flexible on time[3].

The **CounterProposalGeneration** function is used by the contractor to generate a counter-proposal in response to an unacceptable proposal. The function works as follows.

[3]Setting the quality request value low does not necessarily result in a speedier search process to find an acceptable solution. A lower value results in the marginal utility cost to decrease; however, it also decreases the marginal utility gain. An acceptable solution should have the marginal utility gain greater than the marginal utility cost.

If there is no previous counter-proposal, the contractor builds the first counter-proposal by removing both the time range and the quality request, and finding the schedule that performs task NL with the minimum marginal utility cost. This counter-proposal has the minimum marginal utility cost because it only respects the contractor agent's constraints and chooses to do the NL task at its most convenient time and in the most convenient way; hence it is more likely to be an acceptable proposal. If a previous counter-proposal exists, the contractor refines the contractee's current proposal by relaxing the time constraints and lowering the quality request alternatively, and this refining process is repeated until an acceptable (MUC < MUG) counter-proposal is found.

Algorithm 5.3.1 Refining process

Related variables: *current proposal (CP): est (earliest start time), dl (deadline), min1 (quality request)*
delt_t1 (=2), delt_t2 (=3): a short period of time;
reduce_ratio (=0.6): a small number used to reduce the minimum quality request of current proposal;

```
begin
   n=0;
   repeat
        n++;
        if((n mod 2) == 1)
           est = est - delt_t1;
           dl = dl + delt_t2;
        else
           minq = minq * reduce_ratio;
        schedules local tasks and NL with new requests (est, dl, minq);
        if a schedule contains NL with all requests satisfied and MUC < MUG
           build the new counter-proposal (NCP) based on this schedule
           (the start time (st) and the finish time (ft) for NL and NL's quality achievement
           are extracted from the schedule and put into a newly created proposal.)
           break;
   until a counter-proposal is built
end
```

The **NewProposalGeneration** function is used by the contractee to build a new proposal based on the contractee's previous proposal and the contractor's current proposal. If the previous proposal is acceptable for the contractor, the current proposal is actually the contractee's previous proposal with detailed implementation information (such as start time, finish time and quality achievement). If the previous proposal is not acceptable, the current proposal is a counter-proposal from the contractor. The contractee performs a two-dimensional search in the time-quality space[4]. As described before, the initial proposal is

[4]The basic idea of this two-dimensional search is a depth-first search: for a given range on the time dimension, the search explores all possible values on the quality dimension; afterwards the search is moved to another range on the time dimension. This algorithm also could be generalized for search on more than two dimensions. The assumption is that the values of each dimension are independent.

built with a time range that maximizes the marginal utility gain. The next new proposal is to search other time areas trying to find a better proposal by reducing marginal utility cost. The initial time range is defined by the earliest start time and the deadline for the NL task. For the non-local task, the earliest start time is the earliest finish time for those tasks (pretasks) that have to be finished before the non-local task can start. The latest finish time is the latest start time for those tasks (without violating their deadline) that have to be performed after the non-local task is finished. The earliest start time can be moved earlier if those pretasks have alternatives that take less time, or part of those pretasks can be dropped without preventing the execution of the NL task. Otherwise, if neither of these two possibilities exist, the earliest start time can't be moved earlier, hence it is unnecessary to search the time area before the initial time range. The latest finish time can be moved later, which can result in additional costs being incurred due to the violation of some later tasks' deadlines (hard or soft deadline), which decreases the marginal utility gain. In this work we assume the earliest start time can't be moved earlier and we only search the time area after the initial time range, but the algorithm could easily be adapted to search in both directions. When the initial proposal is built the contractee agent has no idea how long it takes the contractor agent to perform the NL task and how much quality it can achieve. The counter-proposal provides the information and it can be used to build a new proposal. The following algorithm describes how the new proposal is constructed. If the current quality achievement (qa) is less than the average quality value, the new proposal requests a higher quality and moves the deadline later to make a high-quality performance more likely; if the current quality achievement (qa) is higher than the average quality value and the previous proposal is the initial proposal (remember the initial proposal does not start with the lowest quality request), the new proposal requests a lower quality with the initial time range to see if a better solution exists with the reduced marginal quality cost. Otherwise, the new proposal moves to a later time range by a step size of 5 (the step size can be adjusted)[5], which is about a half of the estimated duration of the non-local task, and requests a lower quality trying to reduce the marginal utility cost. This new proposal is evaluated and if the gain is larger than the estimated cost (it is a good proposal), it is sent to the contractor; otherwise, the proposal is modified to make it closer to the initial proposal so that the gain could be higher. This process is repeated until a good new proposal is found. The above procedure is applied when the previous proposal is acceptable and the current proposal is actually the contractee's previous proposal with the detailed implementation information. When the previous proposal is not acceptable, the current proposal is a counter-proposal from the contractor. The first counter-proposal is built by throwing away all constraints from the contractee and finding the most convenient way to perform the non-local task. In this situation, the contractee agent analyzes why the previous proposal fails; if it fails because the initial time range is too short, it enlarges the range by moving the deadline later and requests a lower quality to see if there is a solution near the initial proposal. Other-

[5]The step size affects the performance of the algorithm in the following way: when the step size is large, it may take less time to find a good solution, but it is also possible to miss some good solutions (for example, when step size is 10, the first range searched is [0, 15], the second range searched should be [10, 25], then the solution that starts at 5 and finishes at 15 could not be found); when the step size is small, it may take longer to find a good solution, but the possibility of missing good solutions is reduced. When the step size is 1, a complete search (in time dimension) is performed.

wise it adjusts the initial range to be a little bit longer than the current execution time and requests a quality higher than the average quality. The second counter-proposal and those counter-proposals that follow it are built by relaxing the previous proposal's request and finding a solution as close to the previous proposal as possible. In this situation, the next proposal is built based on the current proposal, by either requesting a higher quality with a later finish time or moving to the next time range by a step size, depending on how much quality is achieved now.

Algorithm 5.3.2 New proposal generating process

Related variables: *Initial proposal (IP): est0 (earliest start time), dl0 (deadline), minq0 (quality request);*
Previous proposal (PP): est1 (earliest start time), dl1 (deadline), minq1 (quality request);
Current proposal (CP): st (start time), ft (finish time), qa (quality achieved);
current duration = ft - st;
muc: marginal utility cost of current proposal;
delt_t (=7): a short period of time;
step_size (=5): the size of the step moved in time dimension;
average_quality_value: the average quality the non-local task may achieve;
quality_increase_ratio (=1.1): a small number used to increase the current quality request;
cost_reduce_ratio (=0.5): a small number used to reduce the current marginal utility cost;
enlarge_rate (=1.3): a small number used to increase current duration;
quality_reduce_ratio (=0.6): a small number used to reduce the quality request;

```
begin
    if (PP is acceptable)
        if (qa < average_quality_value and not in the initial range)
            est = st;
            dl = ft + delt_t;
            minq = average_quality_value * quality_increase_ratio(1.1);
        else if (qa > average_quality_value and in the initial range)
            est = est1;
            dl = dl1;
            minq = average_quality_value * quality_reduce_ratio(0.6);
            muc = muc * cost_reduce_ratio(0.5);
        else
            est = est1 + step_size;
            dl = est + current_duration;
            minq = average_quality_value * quality_reduce_ratio(0.6);
            muc = muc * cost_reduce_ratio(0.5);
    else
        if (first counter-proposal)
            if(dl1 - est1 < current_duration)
                est = est1;
                dl = est + current_duration * enlarge_rate(1.3);
                minq = average_quality_value * quality_reduce_ratio(0.6);
                muc = muc * cost_reduce_ratio(0.5);
```

```
            else
                est = est1;
                dl = est + current_duration + delt_1;
                minq = average_quality_value * quality_increse_ratio(1.1);
        else
            if (qa > average_quality_value)
                est = st;
                dl = ft + delt_1;;
                minq = average_quality_value * quality_increase_ratio(1.1);
            else
                est = st + step_size;
                dl = est + current_duration;
                minq = average_quality_value * quality_reduce_ratio(0.6);
                muc = muc * cost_reduce_ratio(0.5);
    repeat
        evaluated new proposal with (est, dl, minq, muc)
        if (mug > muc)
            find a good new proposal;
            break;
        else
            move closer to the previous proposal
            if (dl < dl0)
                dl = (dl + dl0)/2;
            else
                dl = est + current_duration + delt_1;
            muc = muc*cost_reduce_rate;
    until a good new proposal is found
end
```

5.3.4 Another Approach - Binary Search

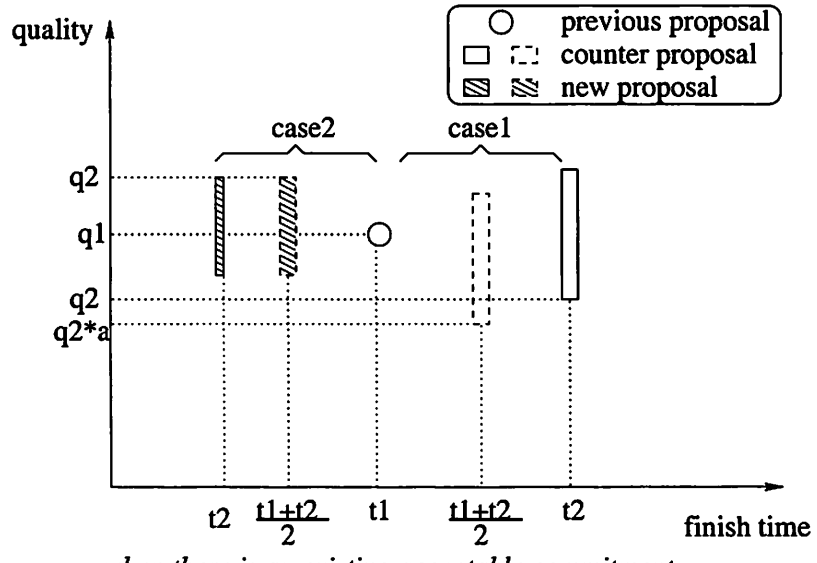
In the previous section we described our algorithm which searches the time dimension range by range, and in each time range, different quality requirements are explored. This algorithm is an approximation of the complete search process; it has a larger search step and uses certain heuristics to control the search process. Earlier, we tried a binary search algorithm [70] whose short description follows. The contractee builds an initial proposal as described above: this initial proposal requests that the non-local task be performed at the most convenient time for the contractee. If the contractor could not accept this proposal, it builds the first counter-proposal using the same procedure as the one described above. Each next proposal from the contractee is a compromise between its own previous proposal and the contractor's counter-proposal, while each next counter-proposal from the contractor is a compromise between the contractee's proposal and the contractor's own previous counter-proposal. Figure 5.2 and Figure 5.3 show how the contractee generates the new proposal based on its previous proposal and the counter-proposal. The contractee agent also behaves differently depending on whether it is trying to improve an existing acceptable commitment

or generating a new proposal in response to a rejection. If there is already an acceptable solution, it tries to find a new solution either with a higher MUG or lower MUC, which will hopefully increase the combined utility. It there is no acceptable solution, it tries to find a solution by relaxing previous request constraints (in quality and/or in time).

Let us see what actions the contractee takes if there is an existing acceptable commitment:

- The contractor informs the contractee that it cannot do task NL as early as the contractee requested. The contractee now asks for a finishing time that is the average of those of the counter proposal and the previous proposal. It also decreases the requested quality at a certain rate (by multiplying it by a value "a" between 0 and 1) thus trying to meet the contractor halfway and with a reduced quality (Figure 5.2, case 1).

- The contractor informs the contractee that it can do task NL as the contractee requested. The contractee asks for a finishing time that is the average of those of the counter proposal and previous proposal and requests the quality that the contractor offered, trying to see if this new pair reduces the contractor's cost and thus increases the combined utility (Figure 5.2, case 2).



when there is an existing acceptable commitment

Figure 5.2: New proposal generation (with acceptable solution)

Let us see what actions the contractee takes if there is no existing acceptable commitment yet:

- The contractor informs the contractee that it cannot do task NL as early as the contractee requested, but it can do it later with a higher quality. The contractee now

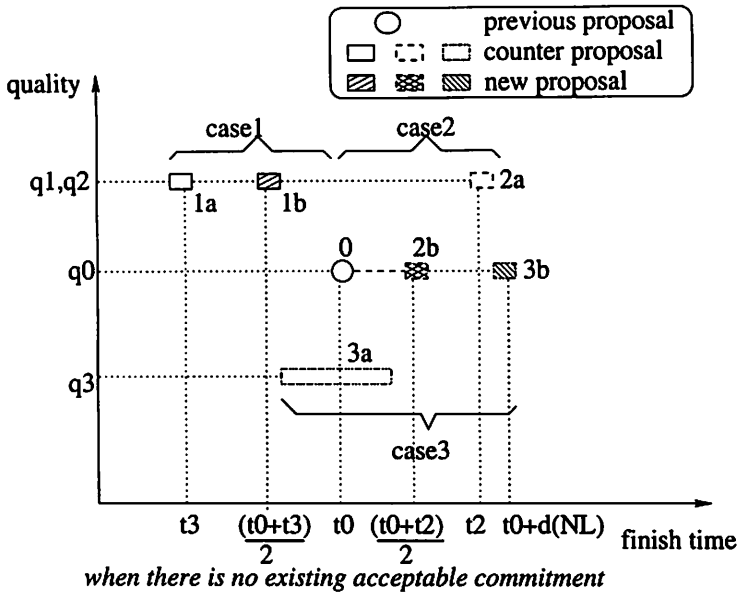


Figure 5.3: New proposal generation (no acceptable solution)

asks for a finishing time that is the average of those of the counter proposal and previous proposal, and it keeps the requested quality the same, thus trying to meet the contractor halfway (Figure 5.3, case 2).

- The contractor informs the contractee that it cannot do task NL as early as the contractee requested and the quality requested is not possible. The contractee asks for a finishing time that is the sum of that of the previous proposal and the duration estimate of task NL, and it keeps the requested quality the same, thus trying to do the task later (Figure 5.3, case 3).

- The contractor informs the contractee that it can do task NL at the requested time or earlier and even with a higher quality than requested. The contractee asks for a finishing time that is the average of those of the counter proposal and the previous proposal and requests the quality that the contractor offered, thus trying to see if this new pair reduces the contractor's cost (Figure 5.3, case 1).

- The contractor informs the contractee that it can do task NL at the requested time or earlier but the quality requested is not possible. The contractee asks for a finishing time that is the sum of that of the previous proposal and the duration estimate of task NL, and it keeps the requested quality the same, thus trying to do the task later (Figure 5.3, case 3).

This binary search algorithm does not work as well as the range-by-range search because it finds fewer acceptable solutions and the quality of those solutions is lower. The following factors contribute to its lower performance:

1. The finish time of the first counter-proposal may not be the actual latest reasonable finish time for the non-local task. The non-local task could be finished later with a

higher quality that may provide a higher combined utility. Since the binary search range is restricted by the finish time of the first counter-proposal, any solution later than that will not be found.

2. The negotiation is about multiple issues, such as the earliest start time, deadline, and quality requirement; hence, the midpoint of the two proposals is difficult to guess based on these three issues. In the implementation, we focus on the deadline (and the finish time) while the earliest start time is adjusted according to the deadline.

3. The domain knowledge used to help the search is incomplete. For example, the duration between the earliest start time and the deadline in the first proposal may be less than the minimum duration of the non-local task's execution by the contractor which causes the failure of the first proposal. When the counter-proposal comes, the minimum duration is available at this time, but it is not used in the rest of the search process.

4. The search process is less structured leading to some solutions often being missed, because both agents are likely to search only in the vicinity of their most favorite proposals.

5.4 Negotiation Protocols and Example

5.4.1 Five Protocols

The negotiation mechanism described in the previous sections serves as a basis for a family of protocol variations differing in the criteria for the negotiation process termination. We examine the following five protocols in this research:

- SingleStep: The contractee sends a proposal commitment to the contractor, the contractor accepts PC if MUG(PC) > MUC(PC); otherwise it rejects PC, and the negotiation is terminated in failure.

- MultiStep-Multiple(n)-Try: The contractee and the contractor perform the negotiation series - "proposal, counter-proposal, new proposal, ... " - until 'n' acceptable solutions with increasing utility gains are found or certain iteration limits are reached. We explore three different values for 'n' in our experiments which are described next.

  - MultiStep-One-Try: MultiStep-Multiple(n)-Try, n=1;
  - MultiStep-Two-Try: MultiStep-Multiple(n)-Try, n=2;
  - MultiStep-Three-Try: MultiStep-Multiple(n)-Try, n=3;

- MultiStep-Limited-Effort: The contractee and the contractor perform the negotiation series - "proposal, counter-proposal, new proposal, ..." - until certain iteration limits are reached. This protocol will explore more possibilities than the above mentioned four protocols when the iteration limit is set to a relatively large number.

Although these protocols differ in the amount of search they do prior to termination, none of them performs a complete search. One reason for that is that generating an optimal local agent schedule for each "what-if" question of the negotiation process is an NP-Hard problem; our scheduler uses heuristics to prune part of the search space and thus not all possible options are expanded. The other reason is that the distributed search space for the possible solutions is also very large and a complete search is too expensive. For example, suppose the earliest start time for the contracted task is 10, the deadline is 30, and the contractor agent has three different approaches to accomplish this task. There would then be a total of 20*3 = 60 possible solutions (starting from time 10, 11, ..., 29 by approach#1, approach#2 or approach#3). And for each possible solution, the agent needs to evaluate it with its other local activities. Thus the computation effort for a complete search is not feasible. Hence, a range-by-range search (with step size of 5) is performed in the whole search space as an approximation of the complete search.

To examine how different protocols work in different situations and to find out the major factors that affect the outcome of negotiation, we have built two agents: the contractee and the contractor. The utility the agent gains by performing task T using schedule S is a multiple attribute utility function, which is a weighted function of the quality achieved, and the cost and duration expended when performing task T.

$$
\begin{aligned}
utility(S) &= quality\_gain(S)*quality\_weight + \\
&\quad cost\_gain(S)*cost\_weight + \\
&\quad duration\_gain(S)*duration\_weight \\
quality\_gain(S) &= \frac{quality(S)}{quality\_threshold} \\
cost\_gain(S) &= \frac{cost\_limit - cost(S)}{cost\_limit} \\
duration\_gain(S) &= \frac{duration\_limit - duration(S)}{duration\_limit}
\end{aligned}
$$

quality(S), cost(S) and *duration(S)* are the quality achieved, cost spent and time spent by schedule S. *quality_threshold, cost_limit, duration_limit, quality_weight, cost_weight* and *duration_weight* are defined in the agent's criteria function. The first three values specify the quality the agent wants to achieve from this task, the cost and the time it wants to expend on this task; the other three values specify the relative importance of the quality, cost and duration attributes[65].

5.4.2 Example

In this section, we use an example to explain how the negotiation mechanism works.

For instance, the contractee is working on task TCE (Figure 5.4). TCE has two subtasks, Task1 and Task2. Task1 has three subtasks, M1, M2 and M3. Each of them takes 9 units of processing time (d:9), has a cost of 10 (c:10) and generates 10 units of quality (q:10). The "sum" associated with a task means the quality of the task is the sum of all its subtasks. Task2 has two subtasks, M4 and M5. There is an "enables" relationship between

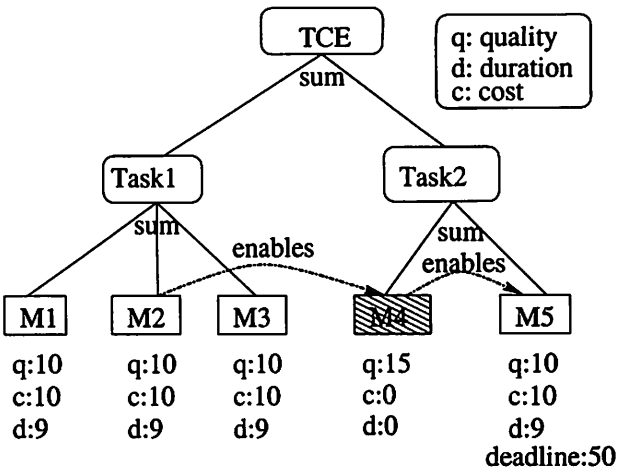


Figure 5.4: The contractee's task structure

M2 and M4, which denotes that M4 can only be started after M2 has been successfully finished. Likewise, another "enables" relationship between M4 and M5 specifies that M5 has to be performed after M4. The deadline constraint associated with M5 indicates it has to be finished by time 50. Subtask M4 is a task that needs to be assigned to another agent (suppose the problem solver makes this decision). The contractor is an agent that could potentially perform task M4. (There could be more than one potential agent. For clarity we only show one). Similarly, Figure 5.4 shows the contractor's local task TCR (the left part of the figure).

In this example, the contractee has the following criteria definition: *quality_threshold* = 50, *cost_limit* = 50, *duration_limit* = 55, *quality_weight* = 0.7, *cost_weight* = 0.15 and *duration_weight* = 0.15. The contractor has a slightly different set of criteria: *quality_thre shold*= 50, *cost_limit* = 50, *duration_limit* = 55, *quality_weight* = 0.7, *cost_weight* = 0.2 and *duration_weight* = 0.1.

Step1: Build-Proposal (Action A in Figure 5.1) The contractee schedules local task structure TCE assuming M4 is not to be done and gets the following schedule S1:
$S1 : M2(0-9)M3(9-18)M1(18-27)$
$Quality(S1) = 30; Cost(S1) = 30; Duration(S1) = 27; Utility(S1) = 0.556$
. Then it schedules TCE assuming that another agent could perform M4 and gets schedule S2:
$S2 : M2(0-9)M3(9-18)M1(18-27)M4[27-27]M5(27-36)$
(with M4's result available at time 27)
$Quality(S2) = 55; Cost(S2) = 40; Duration(S2) = 36; Utility(S2) = 0.8518$
. Then it builds the commitment PC0 based on S2: since Method2 enables Method4, the earliest start time is 9; the deadline is 27 because it has to be finished before Method5's scheduled start time 27; the given range 18 here seems very flexible compared to the estimated duration(10.5), so the quality request is set to a higher value(18.0) than the average value(15.0) of the estimation quality achievement.

PC0: [M4, earliest_start_time: 9, latest_finish_time: 27, quality_request: 18]
$MUG(M4) = Utility(S2) - Utility(S1) = 0.8518 - 0.556 = 0.295$

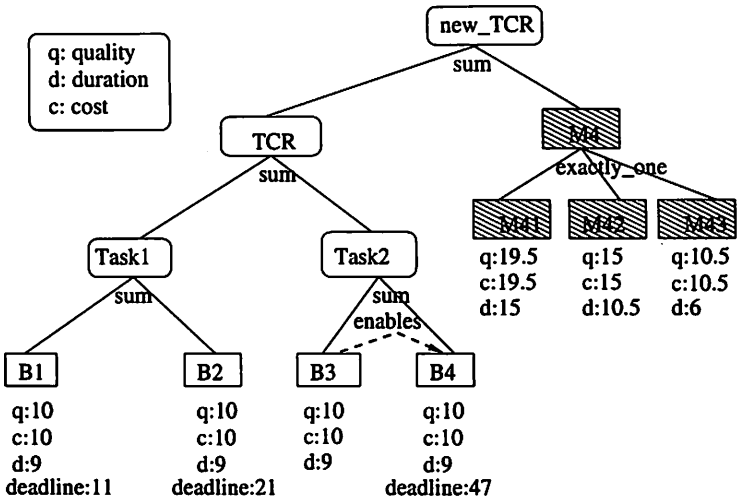


Figure 5.5: The contractor's task structure

Step2: Evaluate-Proposal (Action J in Figure 5.1) The contractor receives this commitment, adds M4 to its local task structure TCR and gets a new task structure new_TCR (Figure 5.5). The contractor instantiates M4 and finds three different plans to perform M4: M41, M42 and M43. Each plan has different quality, cost and duration characteristics. These three choices are represented as three subtasks of M4 with "exactly_ one" quality accumulation function (qaf) in the TÆMS structure.
The contractor schedules new_TCR with PC0:[M4, earliest_start_time: 9, latest_finish _time: 27, quality_request: 18], and finds the following schedule S3:
$S3 : B2(0-9)M41[9-24]B1(24-33)B4(33-42)$
Quality(S3)=30[6]; Cost(S3)=49.5; Duration(S3)=42; Utility(S3)=0.446
Compared to the schedule S4 without performing Task M4:
$S4 : B1(0-9)B2(9-18)B3(18-27)B4(27-36)$
$Quality(S4) = 40; Cost(S4) = 40; Duration(S4) = 36; Utility(S4) = 0.635$
the marginal utility cost is Utility(S4) - Utility(S3) = 0.189. Then it sends the following information back to the contractee agent:
PC0 [M4, start_time: 9, finish_time: 24, quality_achieved: 19.5]
$MUC(PC0) = Utility(S4) - Utility(S3) = 0.189.$

Step3: Re-Evaluate-Proposal (Action I in Figure 5.1) The contractee receives PC0 and re-evaluates it since it received a higher quality and the earlier finish time than it requested:

[6]Notice the quality of schedule S3 does not include the quality achieved by M41 since it does not contribute to the contractor's local utility.

PC0 [M4, start_time: 9, finish_time: 24, quality_achieved: 19.5]
$MUG(PC0) = 0.358 > MUC(PC0) = 0.189$
So this is an acceptable commitment. In either a SingleStep protocol or a MultiStep-One-Try protocol, the contractee stops here and accepts PC0 with the combined utility gain of 0.169. In a MultiStep-Two-Try or a MultiStep-Three-Try protocol, the contractee continues negotiation and tries to find a better commitment.

Step4: Generate-New-Proposal (Action D in Figure 5.1) If the contractee decides to find another solution, it attempts to improve the proposal based on its previous proposal and the current proposal from the contractor. It constructs a new proposal by decreasing the quality request, as the algorithm described in Section 5.3.3:
PC1 [M4, earliest_start_time: 9, latest_finish_time: 27, quality_request: 13.5]
The contractee agent evaluates this new proposal and finds schedule S5 with this commitment.
$S5 : M2(0-9)M3(9-18)M1(18-27)M4[27-27]M5(27-36)$
(with M4's result available at 27 and achieved quality of 13.5)
$Utility(S5) = 0.83, MUG(PC1) = 0.274$
PC1 is sent to the contractor.

Step5: Evaluate-Proposal (Action J in Figure 5.1) The contractor finds schedule S6 that satisfies the commitment PC1.
$S6 : B2(0-9)M42[9-19]B3(19-28)B4(28-37)$
$Quality(S6) = 30; Cost(S6) = 45; Duration(S6) = 37; Utility(S6) = 0.472,$
$MUC(PC1) = Utility(S3) - Utility(S6) = 0.635 - 0.472 = 0.163$
Since the marginal gain is greater than the cost, PC1 is acceptable.

Step6: Re-Evaluate-Proposal (Action I in Figure 5.1) The contractee receives PC1 and re-evaluates it based on the higher quality and the earlier than requested finish time it gets:
PC1 [M4, start_time: 9, finish_time: 19, quality_request: 15]
$MUG(PC1) = 0.295 > MUC(PC1) = 0.163$
so this is an acceptable commitment. However this commitment with the combined utility gain of 0.132 is worse than the first solution, so in a MultiStep-Two-Try protocol or a MultiStep-Three-Try protocol. The contractee continues negotiation and tries to find a better commitment.

Step7: Generate-New-Proposal (Action D in Figure 5.1) It rebuilds a new proposal by moving the earliest start time later, from old start time 9 to 14 by adding 5 (the step size is 5), as the algorithm described in Section 5.3.3:
PC2 [M4, earliest_start_time:14, latest_finish_time: 27, quality_request: 9.0]
The contractee agent evaluates this new proposal and finds schedule S7 with this commitment.
$S7 : M2(0-9)M3(9-18)M1(18-27)M4[27-27]M5(27-36)$
(with M4's result available at 27 and achieved quality of 9.0)

$Utility(S7) = 0.767, MUG(PC2) = 0.211$

PC2 is sent to the contractor.

Step8: Evaluate-Proposal (Action J in Figure 5.1) The contractor finds schedule S8 that satisfies the commitment PC2.

$S8 : B1(0-9)B2(9-18)M43[18-24]B3(24-33)B4(33-42)$

$Quality(S8) = 40; Cost(S8) = 50.5; Duration(S8) = 42; Utility(S8) = 0.582,$

$MUC(PC2) = Utility(S3) - Utility(S8) = 0.635 - 0.582 = 0.053$

Since the marginal gain is greater than the cost, PC2 is acceptable.

Step9: Re-Evaluate-Proposal (Action I in Figure 5.1) The contractee receives PC2 and re-evaluates it based on the higher quality and the earlier than requested finish time it gets:

PC2 [M4, start_time:18, finish_time: 24, quality_request: 10.5]

$MUG(PC2) = 0.232 > MUC(PC2) = 0.053$

so this is a better acceptable commitment. In a MultiStep-Two-Try protocol, the contractee agent will stop and accept this commitment with the combined utility gain of 0.179. In a MultiStep-Three-Try protocol, the contractee continues negotiation and tries to find a better commitment.

Step10: Generate-New-Proposal (Action D in Figure 5.1) It rebuilds a new proposal by requesting a higher quality and extending the deadline, as the algorithm described in Section 5.3.3:

PC3 [M4, earliest_start_time:18, latest_finish_time: 31, quality_request: 11.55]

The contractee agent evaluates this new proposal and finds schedule S9 with this commitment.

$S9 : M2(0-9)M3(9-18)M1(18-27)M4[31-31]M5(31-40)$

(with M4's result available at 31 and achieved quality of 11.55)

$Utility(S9) = 0.767, MUG(PC3) = 0.236$

PC3 is sent to the contractor.

Step11: Evaluate-Proposal (Action J in Figure 5.1) The contractor finds schedule S10 that satisfies the commitment PC3.

$S10 : B1(0-9)B2(9-18)M42[18-28]B3(28-37)B4(37-46)$

$Quality(S10) = 40; Cost(S10) = 55; Duration(S10) = 46; Utility(S10) = 0.556$

$MUC(PC2) = Utility(S3) - Utility(S10) = 0.635 - 0.556 = 0.079$

Since the marginal gain is greater than the cost, PC3 is acceptable.

Step12: Re-Evaluate-Proposal (Action I in Figure 5.1) The contractee receives PC3 and re-evaluates it based on the higher quality and the earlier than requested finish time it gets:

PC3 [M4, start_time:18, finish_time: 28, quality_request: 15]

$MUG(PC3) = 0.293 > MUC(PC3) = 0.079$

So PC3 with the combined utility gain of 0.214 is the best solution found so far. In a MultiStep-Three-Try protocol, the contractee agent will accept this commitment and stop; in a MultiStep-Limited-Search protocol, if the predefined iteration limits have not been reached, the agent will continue searching.

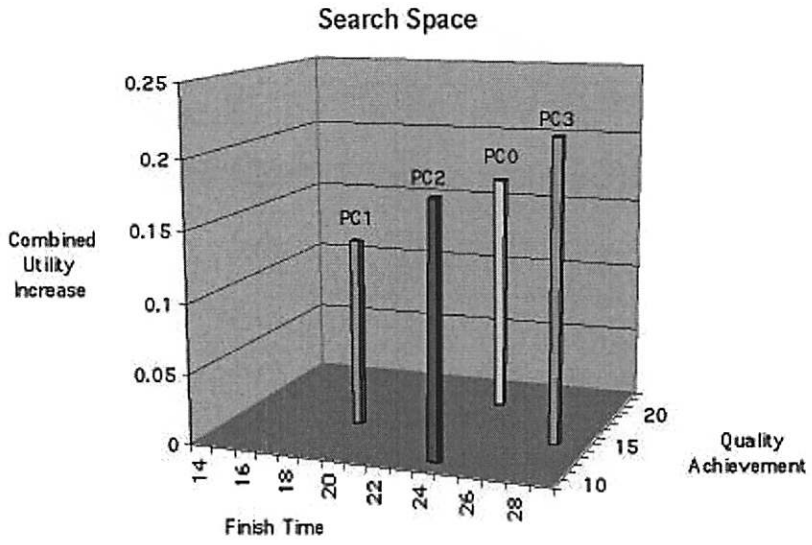


Figure 5.6: Search space

By now, the contractee has obtained four acceptable commitments: PC0 starts from 9 and finishes at 24, achieves quality 19.5 and has a combined utility increment of 0.169, PC1 starts from 9 and finishes at 19, achieves quality 15 and has a combined utility increment of 0.133, PC2 starts from 18 and finishes at 24, achieves quality 10.5 and has a combined utility increment of 0.179, PC3 starts from 18 and finishes at 28, achieves quality 15 and has a combined utility increment of 0.214. PC3 is the best solution. Figure 5.6 shows PC, PC1, PC2 and PC3 in the 2-dimensional search space.

5.5 Experiment & Evaluation

The experiment is designed to examine how different protocols work in different situations and find what major factors affect the negotiation outcomes. Two agents have been constructed, the contractee agent and the contractor agent. Each agent sequentially processes a set of different task structures. Each task structure is generated as a variant of the basic task structure shown in Figure 5.4 and Figure 5.5. Although the basic task structure does not change in the experiments, it represents a set of problems where the negotiation occurs over a nonlocal task which has interrelationships with other local tasks, and both the contractee agent and the contractor agent deal with complex local tasks which carry temporal constraints and interrelationships among them. The number of temporal constraints

(deadline and earliest start time) attached to a task varies from 0 to 3, and the number of "enables" interrelationships among tasks varies from 0 to 3. For example, in Figure 5.4, there is a deadline constraint attached to task M2, and there is an "enables" interrelationship between M4 and M5. The purpose is to generate negotiation contexts with different difficulties. There is a total of 4096 ($2^6 * 2^6$) test cases obtained from the combinations of these task structures. Figure 5.7 shows the contractee's task structure and the contractor's task structure with all six possible temporal constraints and all six possible interrelationships. Besides the five different protocols described in section 5.4.1, we also developed a "complete search" algorithm as a comparison base for the experiment. The "complete search" algorithm searches each start time point and finish time point in a reasonable time range, combined with each possible approach for the non-local task. This "complete search" however is still not guaranteed to find the optimal solution since the scheduler we use is itself heuristic and does not always find the optimal local schedule for the given constraints. Although the local scheduling is still not "complete", both agents explore all generated possibilities and find which solution has the highest combined utility; such a solution is called the "best solution". We compare the solution from each protocol to the best solution to evaluate its effectiveness.

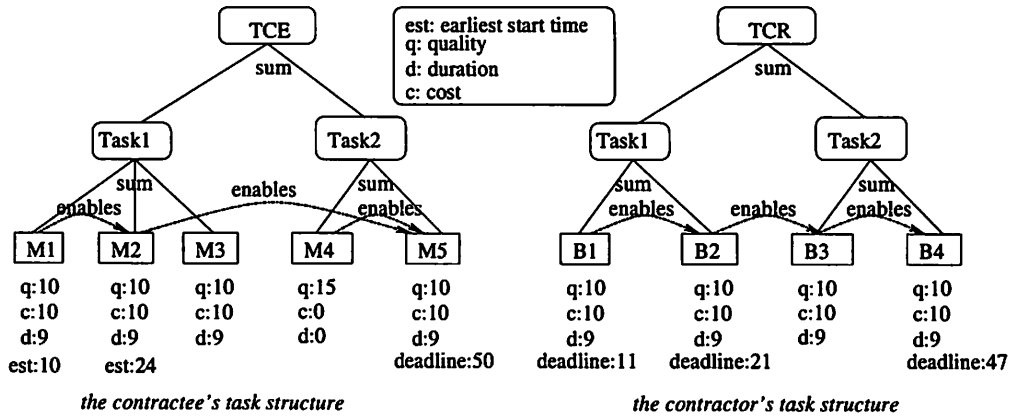


Figure 5.7: Examples of various task structures

We collect the following data for each test case in the experiment:

- *Outcome* (Success/Fail): A negotiation session is successful if it ends with a commitment that increases the combined utility. Otherwise it fails.
- *Utility Gain*: The difference between the MUG(C) and MUC(C). C is the finally adopted commitment. If the negotiation session fails, Utility Gain is 0.
- *Gain Percentage*: The percentage of the utility gain out of the combined utility without performing the task allocation.
- *Solution Quality*: How good this solution is compared to the best solution from the "complete search". We compare only the utility increase from the negotiation. Suppose a negotiation solution results in the combined utility increased by 18%, and the

best negotiation solution could increase the combined utility by 20%, then the quality of this negotiation solution is 90% (= 18/20*100%). If a negotiation fails without reaching an agreement, the quality of the solution is defined as 0.

- *Complexity of Task Structures*: The number of constraints ("deadline" and "enables" relationships) in the task structures that are mapped onto the complexity measure of the negotiation. The formula we use to calculate the complexity is as follows:
$complexity = ir1 + tc1 + ir2 + tc2 + \frac{ir1*tc1+ir2*tc2+ir1*ir2+tc1*tc2+ir1*tc2+ir2*tc1}{6}$
$ir1$: number of interrelationships in the contractee's task structure; $tc1$: number of temporal constraints in the contractee's task structure; $ir2$: number of interrelationships in the contractor's task structure; $tc2$: number of temporal constraints in the contractor's task structure;
For example, in Figure 5.7, $ir1 = 3, tc1 = 3, ir2 = 3, tc2 = 3, complexity = 21$. This formula is based on the idea that the more constraints there are, the more complicated the task structures are, and the more difficult the negotiation would be.

- *Number of Negotiation Steps*: The length of the negotiation series (Proposal[1] - Counter-Proposal[2]- Proposal[3] - Counter-Proposal[4] - ...).

Table 5.1: Comparison of five protocols (AGP: the average of the gain percentage over all cases. ANNS: the average number of the negotiation steps over all the cases. GPS: the negotiation gain over each step (GPS=AGP/ANNS). SQ: the average of the solution quality over all cases.)

	Success	AGP	ANNS	GPS	SQ
SingleStep	2850	7.63	1.0	7.63	51.44
MultiStep-One-Try	4088	10.17	1.48	6.87	72.37
MultiStep-Two-Try	4088	11.9	4.69	2.55	84.57
MultiStep-Three-Try	4088	13.4	6.42	2.09	96.21
MultiStep-Limited-Effort	4088	13.9	8.15	1.7	99.36

Table 5.1 shows the comparison of these five protocols. Out of the 4096 test cases, the SingleStep protocol succeeds in 2850 cases, the other four MultiStep protocols succeed in 4088 cases. Among these 4088 cases, there are 1508 cases in which the MultiStep-One-Try protocol finds a better solution than the SingleStep protocol; there are 2298 cases in which the MultiStep-Two-Try protocol finds a better solution than the MultiStep-One-Try protocol; there are 2168 cases in which the MultiStep-Three-Try protocol finds a better solution than the MultiStep-Two-Try protocol; there are 675 cases in which the MultiStep-Limited-Effort protocol finds a better solution than the MultiStep-Three-Try protocol. For the SingleStep protocol, the average solution quality(SQ) is 51.44% of the best solution; the average number of the negotiation steps(ANNS) is 1, and the average utility gain from negotiation (AGP) is 7.63% of the combined utility without negotiation. Hence the average negotiation gain over each negotiation step (GPS=AGP/ANNS)) is 7.63% of the combined

utility without negotiation. For the four MultiStep protocols, as the average negotiation step number (ANNS) increases from 1.48 to 8.15, the average solution quality(SQ) also increases from 72.37% to 99.36%, while the negotiation gain over each step (GPS) decreases from 6.87% to 1.7%.

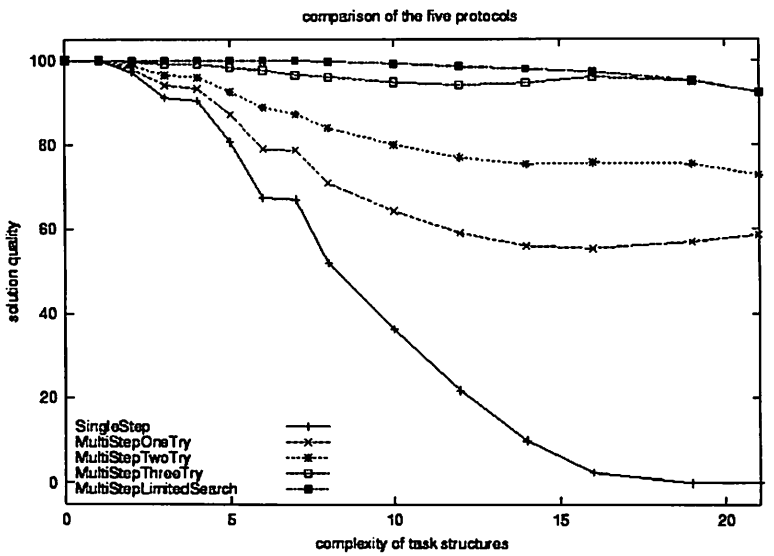


Figure 5.8: Comparison of five protocols according to complexity (the solution quality is a relative quality compared to the best solution, number 100 means the best solution)

Figure 5.8 shows how these five protocols perform when the complexity of the task structures changes. As the complexity of the task structures increases, the negotiation problem becomes harder to solve, because the search space for a potentially valid solution is narrowed as the number of constraints in the task structures grows. The SingleStep protocol performs almost as well as the other four protocols when the problem is very easy (the complexity is very low), but its performance decreases dramatically as the complexity increases. Furthermore, Figure 5.8 tells us that the MultiStep-$(n+1)$-Try protocol performs much better than the MultiStep-n-Try protocol in the more constrained situation (e.g. when complexity is larger than 5). When there are fewer constraints or too many constraints, the extra search beyond the MultiStep-Three-Try does not bring additional gains. This is because when there are fewer constraints it is very likely that the previous search has found a very good solution; and when there are many constraints, it is hard to find a better solution as a result of the extra search.

Figure 5.9 shows the performance of each protocol with error bar (the confidence level is 0.9). We find that as the negotiation effort increases, the performance of the protocol is more stable. SingleStep protocol sometimes fails even in the medium complexity situation, MultiStep-One-Try protocol sometimes has a solution quality that is only 10% of the best

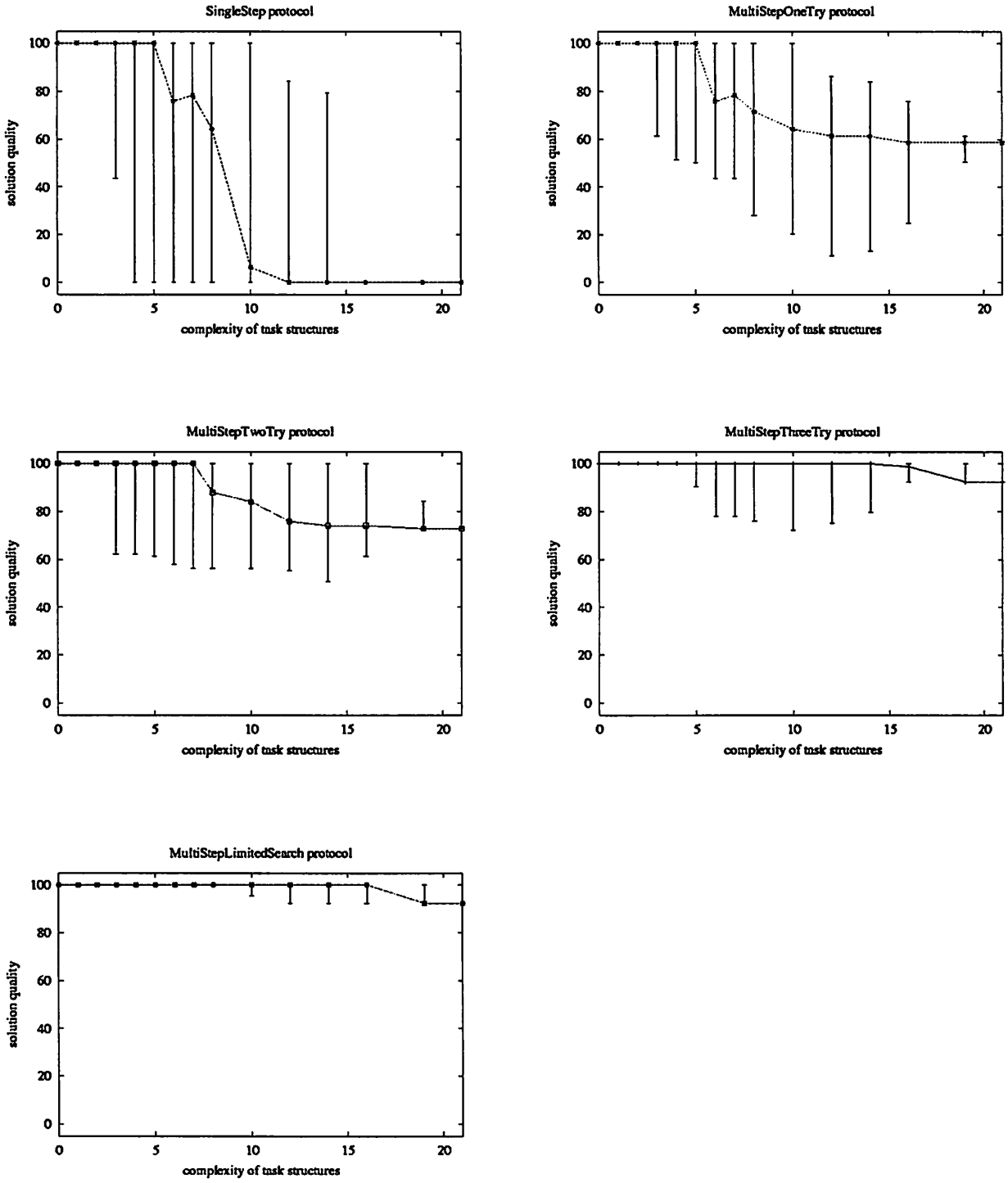


Figure 5.9: The performance of all protocols with error bars

solution. However, with MultiStep-Two-Try protocol, we have the confidence that over 90% of the time, the solution quality is at least 50% of the best solution. With MultiStep-Three-Try, the lower bound of the solution quality is raised to 70% of the best solution. The agent can choose an appropriate protocol depending on its quality requirement and available time.

The above mentioned data has shown that the performance of each protocol is highly related to the difficulty of the specific negotiation problem. Because each protocol requires different amounts of negotiation effort, it is important for an agent to choose an appropriate protocol that balances the negotiation gain and negotiation effort. Negotiation gain can be represented as the *utility gain* from the negotiation; negotiation effort can be measured by the *number of negotiation steps*. The negotiation effort grows as the *number of negotiation steps* increases. The negotiation cost affects the agent's utility for the following reasons. The first reason is that the negotiation process consumes resources (i.e. time, computational capability, communication capacity, etc.) that otherwise could be used for other tasks; the second reason is that the negotiation process itself has an influence on how and when the contracted task could be executed, which probably could reduce the utility. For example, the contracted task without negotiation could be started as early as time 10; however the negotiation process also starts at time 10. The longer the negotiation process takes, the later the task can actually be started. More generally, the effect of the negotiation cost on the utility is domain dependent. The following domain characteristics are related: how much slack time there is for the contracted task; how much advance time is available for negotiation without affecting the earliest start time of the task; and the frequency of new tasks arriving, opportunity cost and so forth. Given the above factors, it is hard to measure exactly how the negotiation cost affects the agent's utility. We use the following approximated approach: to make the negotiation cost and gain comparable, the *number of negotiation steps (n)* can be mapped into a certain percentage of utility ($c * n$) by multiplying a constant c. The value of c can be chosen according to the actual situation and it should reflect how the negotiation cost affects the overall utility. Without losing generality, c is set to 0.5 in this experiment. That means each step of negotiation decreases the achieved combined utility by 0.5% the initial combined utility without negotiation. The net negotiation gain in Figure 5.10 is calculated as the following formula:

$$net\_negotiation\_gain = \frac{\frac{combined\_utility\_after\_negotiation - initial\_combined\_utility\_without\_negotiation}{initial\_combined\_utility\_without\_negotiation} - c*n}{\frac{best\_solution\_gain}{initial\_combined\_utility\_without\_negotiation}} * 100$$

Figure 5.10 shows the comparison of the *net negotiation gain* (the negotiation gain minus the negotiation effort) of the five protocols. There is a phase transition phenomenon: when the negotiation situation is very simple (complexity < 5), the Single-Step protocol works as well as the MultiStep-One-Try protocol, and the MultiStep-Two-Try protocol and the MultiStep-Three-Try protocol are not good choices. When the negotiation situation is very difficult (complexity > 19), the MultiStep-One-Try protocol should be chosen; the extra negotiation effort of the MultiStep-Two-Try and the MultiStep-Three-Try protocol does not bring reasonable extra gain. When the negotiation situation is of medium difficulty, then the extra gain exceeds the extra effort, and the MultiStep-Three-Try protocol is advantageous in this phase. The MultiStep-Limited-Effort is not a good choice in all kinds of situations since the negotiation cost is too high. The difficulty of the negotiation

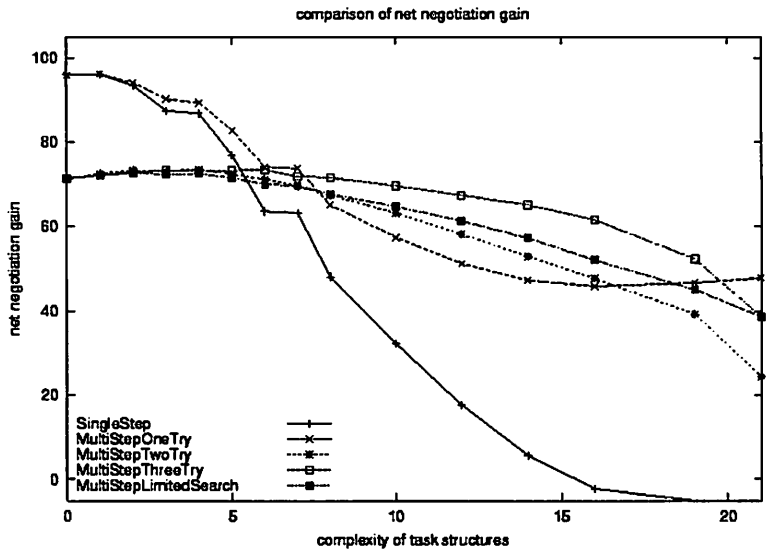


Figure 5.10: Negotiation gain beyond effort

situation is simply measured by the number of constraints in the agents' task structures, which is a "reasonable" measure but by no means a completely accurate measure of the distributed search complexity. The contractor agent can inform the contractee agent of its local constraints number before the negotiation process starts, and the contract agent can decide which protocol to choose (how much effort to put in the negotiation) according to the estimate of the negotiation difficulty.

complexity	< 5.0			5.0 - 19.0				>=19.0		
	AGP	ANNS	GPS	AGP	ANNS	GPS		AGP	ANNS	GPS
SingleStep	12.81	1	12.81	7.24	1	7.24		0	1	0
MultiStep-One-Try	13.07	1.05	12.45	9.96	1.51	6.59		5.8	2	2.9
MultiStep-Two-Try	13.3	6.14	2.16	11.85	4.57	2.6		7.89	7.23	1.09
MultiStep-Three-Try	13.58	6.83	2.2	13.4	6.38	2.1		9.73	8.62	1.13
MultiStep-Limited-Effort	13.67	7.15	2.22	13.93	8.23	1.69		9.74	9.77	0.997
utility/negotiation-step	< 0.20	< 5.19	> 5.19	< 0.29	< 0.78	< 5.31	> 5.31	< 0.39	< 5.46	> 5.46
SingleStep			best				best			best
MultiStep-One-Try		best				best			best	
MultiStep-Two-Try										
MultiStep-Three-Try					best					
MultiStep-Limited-Effort	best			best				best		

Figure 5.11: Comparison of five protocols (AGP: the average of the gain percentage. ANNS: the average number of the negotiation steps. GPS: the negotiation gain over each step.)

Figure 5.11 shows each protocol's performance under the three different situations: low complexity, medium complexity and high complexity. The table shows how much the negotiation gain (AGP) is and how much the negotiation cost (ANNS) is. The negotiation

gain over each step (GPS) provides an upper bound limit on the usefulness of the protocol (cost is less than gain). Let us assume that in a specified application domain, each negotiation step costs c percent of overall utility, then if c is less than GPS the protocol is useful. For example, in the low complexity situation, the MultiStep-Two-Try protocol could be useful only if each negotiation step costs less than 2.16% of overall utility. Also the table shows under each of those three situations, which protocol is the best one when the negotiation cost changes. For example, in the situation of medium complexity (complexity between 5 and 19), when each negotiation step costs less than 0.29% of overall utility, the MultiStep-Limited-Effort protocol is the best; when each negotiation step costs more than 0.29% but less than 0.78% of overall utility, the MultiStep-Three-Try protocol is the best; when each negotiation step costs more than 0.78% but less than 5.31% of overall utility, the MultiStep-One-Try protocol is the best; when each negotiation step costs more than 5.31% of overall utility, the SingleStep protocol is the best. Figure 5.12 illustrates the above idea in another way.

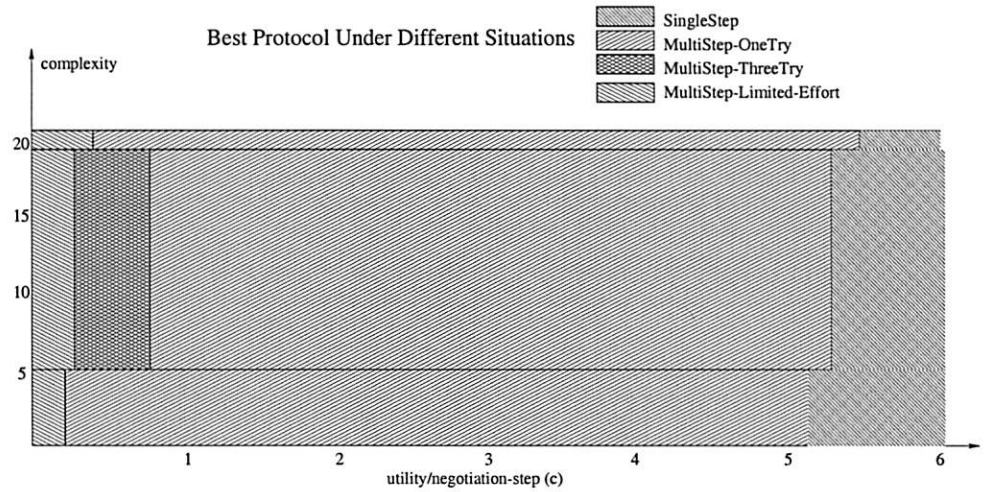


Figure 5.12: Best protocol under different situations

Based on the above mentioned empirical results, the following observations can be made:

1. In almost all the situations (except the very simple situation), the MultiStep-One-Try protocol is much better than the SingleStep protocol, since it achieves much more gain with less extra effort.

2. The MultiStep-Two-Try and MultiStep-Three-Try protocols are worthwhile in the medium-difficult negotiation situation. The agent could decide if it is worthwhile to spend any extra effort. If the task structures have very few or very tight constraints then the MultiStep-One-Try protocol is sufficient.

3. The complexity measure based on the number of constraints can be used to choose the appropriate protocol that balances the negotiation gain and negotiation effort.

5.6 Additional Thoughts

As mentioned in the previous section, the protocols' performance is highly related to the difficulty of the negotiation problem; we tried to find some good predictors that the agent can use to select the best protocol. The complexity measure is such a predictor; it is easily obtained (just count the number of constraints) and it works well as shown in section 5.5. However, it is not a perfect measure. The large variance in the data shows that it can't capture more detailed information. Hence, we have done additional work trying to find a better predictor.

5.6.1 Analysis of Solution Space

First, we tried to understand what makes a negotiation problem difficult. The structure of the solution space was examined because we thought the number of solutions may affect the difficulty of a problem. A complete search was performed and all solutions were obtained (of course, this measure could not be a predictor, but just helps us analyze the problem). The following four measures were calculated:

1. Solution Number (sn): the number of all solutions.
2. Unique Solution Number (usn): the number of unique solution. This does not count different approaches at the same start time.
3. Good Solution Number (gsn): the number of good solutions. If the solution is better than 3/4 of the best solution, it is a good solution.
4. Unique Good Solution Number (ugsn): the number of unique good solutions.

The relationship between the performance of the protocols and these measures was then studied. It needs to be pointed out that the difficulty of the negotiation problem is actually an abstract term. In general, an easy problem should be easy to solve, a hard problem is hard to solve. However, how hard or how easy to solve a problem also depends on what algorithm is used to solve it. Hence, the performance of the negotiation protocols may not reflect the difficulty of the problem perfectly, but it does give us some sense of the difficulty.

All four of these measures provide a similar result. Let's use the solution number as an example to explain our observation (Figure 5.13). It seems that the difficulty is not changing continuously according to the solution number. The solution number divides the problems into two categories: When the solution number is smaller than a certain number (85), the SingleStep protocol always fails, and there is a relatively big gap between the performance of the first two MultiStep protocols and the other two MultiStep protocols. When the solution number is larger than a certain number (85), there is no continuous large gap among all those protocols, although the SingleStep protocol still fails at certain points.

From the above observation we find that it is not only the solution number that affects how each protocol performs, but also some other characteristics of the solution space structure, such as how the solutions are distributed in the search space, and how far away the best solution (there may be more than one best solution) is from the initial proposal, etc.

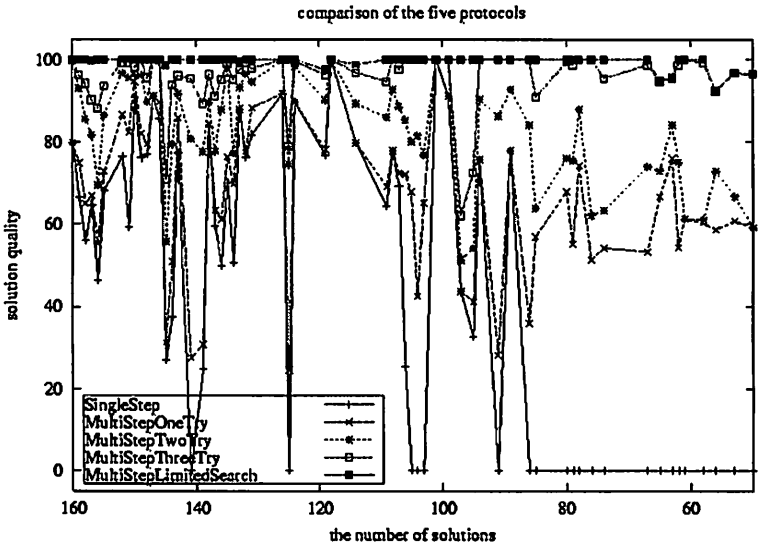


Figure 5.13: Comparison based on solution number

5.6.2 Definition of Flexibility

We also developed a more complex flexibility measure (extending the complexity measure described by Deshmukh in [17]) that measures how flexible a set of tasks is:

A set of tasks $TS = \{T_1, T_2, \ldots, T_n\}$

For each task T_i: est_i (earliest start time), dl_i (deadline), d_i (duration of processing time)

For any two tasks T_i and T_j: if $p(i,j) = 1$, then task T_i must be finished before task T_j; otherwise they can be performed in any order.

$\varphi_i = \frac{d_i}{\sum_{i=1}^{n} d_i}$

if $p(i,j) = 1$, $\pi_{ij} = dl_i - est_i - p_i$, the slack time of the task;

if $p(i,j) = 0$, $\pi_{ij} = 0$;

$\pi^*_{ij} = \pi_{ij} * \varphi_i$

The flexibility of a set of tasks TS is defined as: $F(TS) = -\sum \pi^*_{ij} * \log \pi^*_{ij}$

If there is no feasible linear schedule for the task group, the flexibility is defined as -1.

The flexibility of a single task t is defined as: $F(t) = \frac{dl(t) - est(t)}{d(t)}$.

The characteristic of the flexibility measure is: the more slack time the tasks have, and the fewer precedence relationships among those tasks, the greater the flexibility is. If the agent has more slack time for its local tasks and fewer precedence constraints among them, it is easier to arrange its local tasks and hence leave more space for additional tasks. The flexibility measure should capture more detailed information about local tasks compared to the complexity measure.

5.6.3 Flexibility Related Measures

Based on the flexibility of a set of tasks and the flexibility of a single task, we developed the following measures to describe a negotiation situation: The contractee has a task structure (TSa) and the contractor has a task structure (TSb), and the contractee needs to assign the non-local task Tnl to the contractor.

1. Flexibility Sum Measure (fsum): fsum = F(TSa) + F(TSb)
2. Flexibility Product Measure (fproduct): fproduct = F(TSa)* F(TSb)
3. Flexibility Max Measure (fmax): fmax = Max (F(TSa), F(TSb))
4. Flexibility Min Measure (fmin): fmin = Min (F(TSa), F(TSb))
5. Task Flexibility Sum Measure (tfsum): tfsum = F(Tnl)+F(TSb)
6. Task Flexibility Product Measure (tfproduct): tfproduct = F(Tnl)*F(TSb)
7. Task Flexibility Max Measure (tfmax): tfmax = Max(F(Tnl), F(TSb))
8. Task Flexibility Min Measure (tfmin): tfmin = Min(F(Tnl), F(TSb))

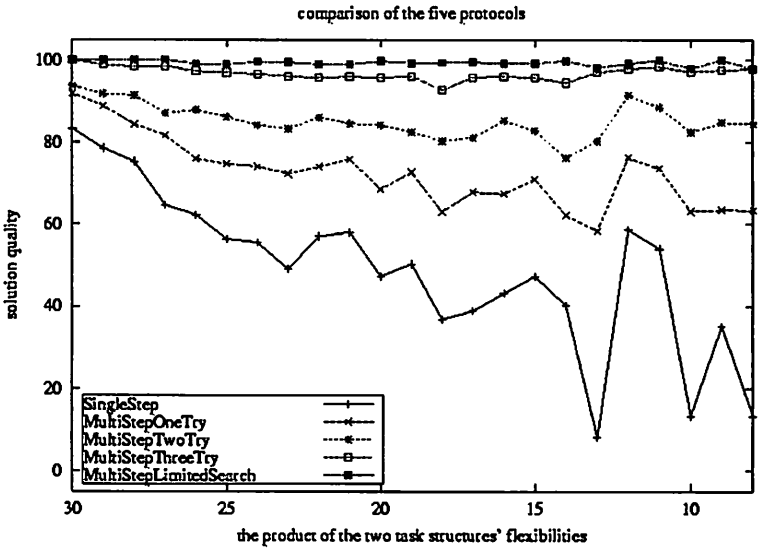


Figure 5.14: Comparison based on flexibility product measure (fproduct)

Almost all these measures (except the fmin measure, since it seems it is not to be a good approach to combine two flexibility measures together) provide results similar to the complexity measure: as the flexibility related measure decreases, the difference among the performance of those protocols increases. Figure 5.14 and 5.15 show the comparison of

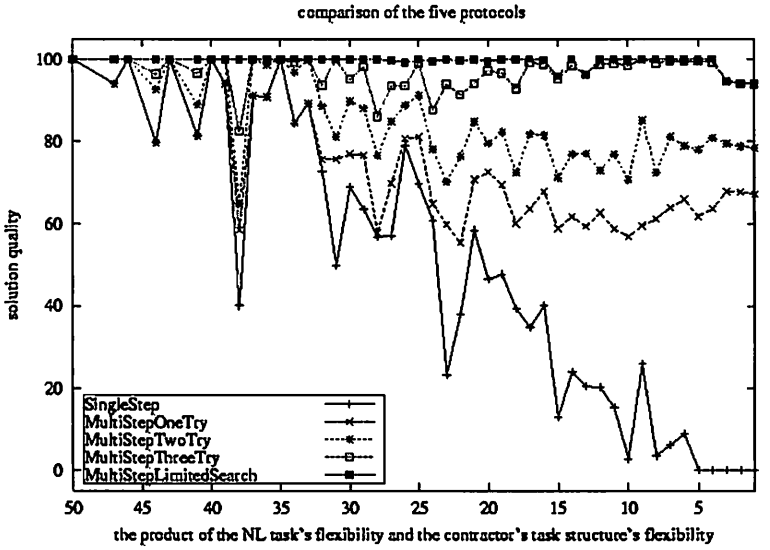


Figure 5.15: Comparison based on task flexibility product measure (tfproduct)

the five protocols according to the $fproduct$ and the $tfproduct$ measures. Why couldn't the flexibility related measures work better? One reason is that the flexibility measure only deals with the slack time that is calculated from the hard deadlines; thus it can't reflect the soft deadline idea [7].

5.6.4 Initial Range Related Measures

We also found that the initial proposed range for the non-local task is an important factor that affects the protocols' performance. The following are two measures based on the initial range:

1. Initial Range and Constraints Number Measure (rconst): rconst = (8- Initial_Range(Tnl)/5) + ir2+tc2 [8]

2. Initial Range and Flexibility Measure (rflex): rflex = Initial_Range(Tnl)/5 + F(TSb)*10

These two measures also provide results similar to the complexity measure as shown in Figure 5.16 and 5.17.

We have tried all the above approaches but did not find a completely satisfying measure of the negotiation difficulty. The problem is much harder than we originally thought. The question of how to combine the characteristics of the subproblems together to predict the

[7]The hard deadline means that the task produces zero utility if it could be be finished before the deadline; while the soft deadline allows a task to generate certain quality even it is finished after the deadline. However the earlier the tasks are finished, the higher the utility could be (see definition of utility function in section 5.4.1).

[8]ir2: number of interrelationship in TSb; tc2: number of temporal constraints in TSb.

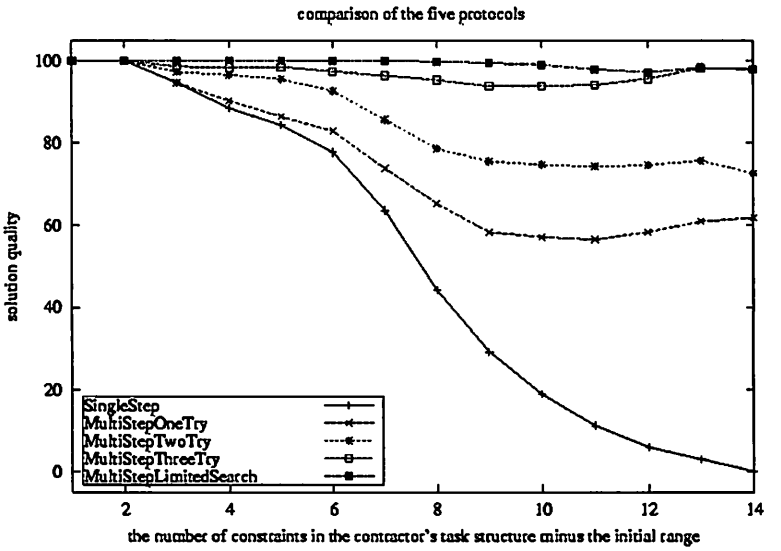


Figure 5.16: Comparison based on initial range and constraints number measure (rconst)

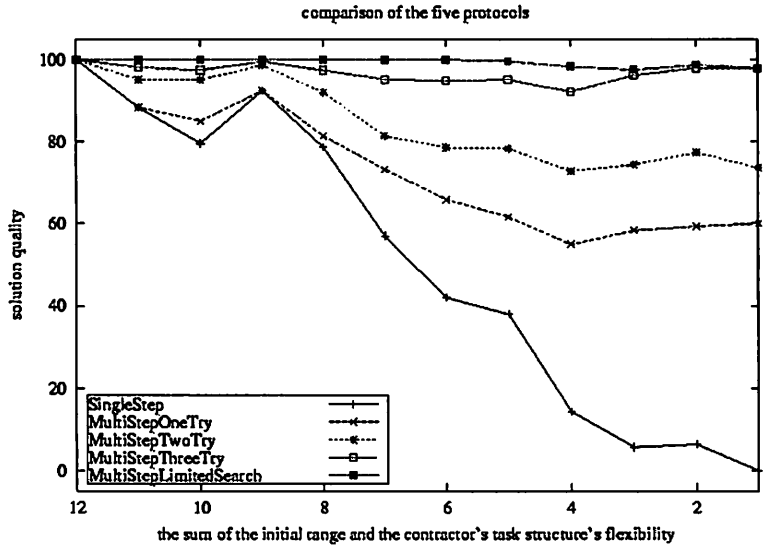


Figure 5.17: Comparison based on initial range and flexibility measure (rflex)

characteristics of the problem as a whole is still an open issue. In hind sight, given that we are doing a distributed optimization search over interdependent search spaces where there can be complicated interdependence among the spaces, this conclusion is not unexpected.

5.7 Different Negotiation Approaches in Retrospect

In Chapter 2, we presented a negotiation process for task contracting between agents. On the other hand, we discussed another negotiation approach for task allocation between cooperative agents. In this section in this chapter, we will compare and contrast these two different negotiation approaches and look at a number of basic questions concerning the negotiation process.

Before we start, there are some points we need to clarify. First, Chapter 2 is not simply another approach to the task allocation problem presented in this chapter. The major focus of Chapter 2 is to solve the multi-linked negotiation problem, which is not addressed in this chapter, where we assume only one issue is under negotiation at a time. Secondly, in this chapter, the negotiation is performed between cooperative agents, and the purpose of negotiation is to find a solution which increases their joint utility. Whereas in Chapter 2, the agents which perform the negotiation are not necessarily cooperative, and the purpose of the negotiation is to find a mutually acceptable commitment.

5.7.1 DTC Scheduler and Partial Order Scheduler

The DTC scheduler works on a TÆMS task structure and a set of given criteria, in order to provide a set of linear schedules ranked according to the given criteria. The schedule generated by the DTC scheduler specifies a set of primitive tasks (methods) and their start times and finish times.

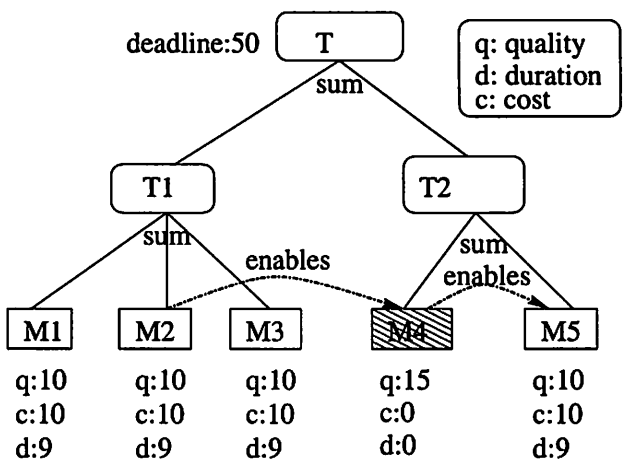


Figure 5.18: An example task structure for task T

For example, given the task structure described in Figure 5.18 and the following criteria definition: *quality_threshold* = 50, *cost_limit* = 50, *duration_limit* = 55, *quality_weight*=

0.7, $cost\_weight = 0.15$ and $duration\_weight = 0.15$, one of the best schedules generated by the DTC schedule is the following:

$S1 : M2(0-9)M3(9-18)M1(18-27)M4[27-27]M5(27-36)$

This schedule $S1$ specifies the start time and finish time of each primitive task. Task $M4$ is a non-local task (which should be contracted to another agent), so it does not consume local process time. By analyzing this schedule and the precedence relationships associated with task $M4$ ($M2 \rightarrow M4 \rightarrow M5$), the feasible time range for $M4$ is found to be $[9, 27]$. This is actually the initial proposed range used in the negotiation which is based on the DTC scheduler (as described in Chapter 5, Section 5.4.2).

The DTC scheduler reasons about the alternative ways to accomplish a task, and finds a linear schedule according to the given criteria. It is a powerful planning tool for an agent to handle complex tasks. However, the linear schedule provided by the DTC scheduler is not oriented to negotiation; hence there are some problems which hinder efficient negotiation.

The first problem is that the linear schedule does not provide the biggest feasible range for a task. For example, the following schedule $S2$ has the same ranking as schedule $S1$:

$S2 : M1(0-9)M2(9-18)M3(18-27)M4[27-27]M5(27-36)$

Based on this schedule, the feasible time range for task $M4$ is found to be $[18, 27]$, which is smaller than it should be. The linear schedule can not guarantee the largest feasible range for a task.

Meanwhile, if we consider the deadline for task T, which is 50, then $M5$ can be delayed as late as $[41, 50]$, which means that the largest feasible range for $M4$ is actually $[9, 41]$. If this range is used for negotiation on $M4$, it can increase the probability that another agent will accept task $M4$.

The second problem is that the linear schedule does not answer "what-if' questions. For example, given that the deadline for task T is 50, if task T is finished before this deadline, the agent will get extra reward. Since the agent needs to reason about the extra reward and the success probability of the negotiation on $M4$, it needs to ask a question such as: what is the largest feasible range for $M4$ if task T is finished by time x ($x <= 50$)? The linear schedule can not provide the answer to this question. A possible way to find the answer is to reschedule the task structure using the DTC scheduler after setting the deadline of task T to x. However, rescheduling using the DTC scheduler involves a non-trivial cost, which is not suitable for the frequently asked "what-if" questions in the negotiation reasoning process. Additionally, as we described in the above example, even after the rescheduling, the linear schedule can not guarantee the correct answer.

Another type of "what-if" question would be: what is the largest feasible range for issue A given the range of issue B? This type of question occurs when there is more than one issue under negotiation.

The above analysis shows that the DTC scheduler is not a powerful tool for efficient negotiation. This is why we need to develop the partial order scheduler to support negotiation-related reasoning. The partial order scheduler works on the linear schedule generated by the DTC scheduler, and does not work directly on the TÆMStask structure. Figure 5.19 shows the partial oder schedule generated from the linear schedule $S1$.

Based on this partial order schedule and using the related algorithms described in Chapter 2.2, the following questions can be answered easily:

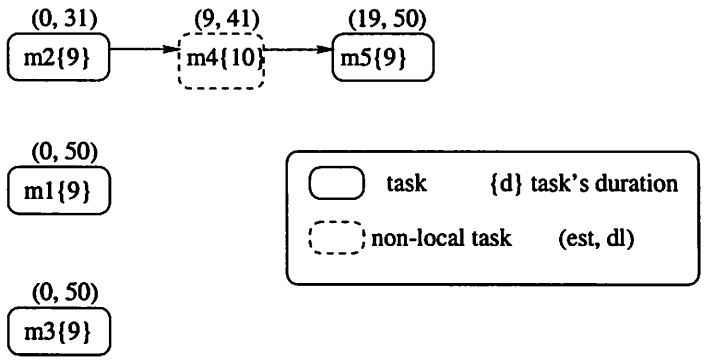


Figure 5.19: A partial order schedule for task T

1. What is the feasible range of a primitive task (method)?
2. How does the finish time of the task affect the feasible range of each method?
3. How does the feasible range of one method affect the feasible ranges of other methods?

The partial order scheduler is built to provide efficient reasoning for negotiation process, which is not offered by the DTC scheduler.

5.7.2 Multi-Step Negotiation and Single-Step Negotiation

In Chapter 5, we presented a multi-step negotiation mechanism, in which agents engage in a series of proposals and counter-offers to decide whether the contractor agent will perform a task for the contractee agent by the specified time with a certain quality. This negotiation mechanism is supported by the DTC scheduler. The agent reschedules its local tasks with the current proposed commitment, in order to find out whether this commitment is acceptable and how good it is.

With the support of the partial order scheduler, the largest feasible range for a task can be found. How does this affect the multi-step negotiation?

If *time* is the only feature in negotiation, and the goal of the negotiation is to find one solution, then the largest feasible range provided by the partial order scheduler can eliminate the needs for a multiple step negotiation process. The contractee agent can use the largest feasible range as the initial proposed range. If the contractor agent can not find a solution during this range, other solutions outside of this range are not acceptable for the contractee agent, so there is no need for further negotiation. If the contractor agent can find a solution during this range, the negotiation finishes successfully because the goal of finding one solution has already been reached.

However, if the agents not only need a solution but also want a "good" solution, a multiple step negotiation is necessary. As we described in this chapter, the agents negotiate to find a solution which increases their joint utility; a better solution therefore provides a greater increase of their joint utility. When the largest feasible range is used as the initial

range, the solution that the contractor agent could find during this range may not be unique. There may be more than one solution in this range, and each solution provides different marginal utility gain for the contractee agent and different marginal utility cost for the contractor agent. More negotiation effort is needed to find a better solution.

When there is more than one feature under negotiation, the number of potential solutions becomes even greater. For example, in the task allocation problem described in Chapter 5, besides the "start time" and "finish time", the "quality achievement" is another feature in negotiation. The contractee agent can specify the largest feasible range and the minimum requirement for quality to be achieved, while the contractor agent can find out whether there is an acceptable solution in a single-step negotiation process. However, if the agents want to find a better solution, they can continue the negotiation after the first solution is found.

For cooperative agents (as described in Chapter 5), a better solution brings a greater increase in their joint utility. For non-cooperative agents, when there is more than one feature under negotiation, it is still possible to find a better solution in which at least one agent's utility increases and no one else's utility decreases, given that the agents may have different preferences on different features. For example, as we discussed in Section 4.3.3, the agents can negotiate over the approach of the task, the type and amount of the transferred MQs, and the reward model of the commitment. Each agent may have different preferences on these features, i.e. the agent which has more uncertainty in its local plan needs more flexibility in the commitment; or one type of MQ looks more attractive to one agent than to others. A broader search over the negotiation space can lead to a better solution for both agents.

In Chapter 2, there are multiple issues under negotiation, and they are related to each other. The feasible range found by the partial order scheduler for one issue is not the largest feasible range; it is constrained by other issues in negotiation. For example, in Figure 5.19, if the agent wants to get some extra reward by finishing task T at time 45 instead of the requested deadline 50, the feasible range available for task $m4$ is $[9, 36]$. Another example is, when task $m5$ is another non-local task that needs to be negotiated, if task $m4$ has the largest feasible range $[9, 41]$, then task $m5$ can only have the smallest feasible range of $[41, 50]$. Since successful negotiation on $m5$ is very difficult given this tight constraint, the agent needs to split the flexibility on these two non-local tasks.

In Chapter 2, a search algorithm is used to find the appropriate value assignment for each feature under negotiation in order to obtain a nearly optimal global solution with the combination of all negotiation results. For a given negotiation solution, the feature assignment specifies a feasible range or a limit for a feature, such as the feasible range $[9, 36]$ for task $m4$, or at most 8 units of MQ transferred for task $m4$. The agent can use this whole range as the proposed range and find out if there is a solution quickly. Another approach is that the agent first proposes part of this range. If a solution is found in this part of the range, the agent can save some flexibility for itself; otherwise, it can try the other part of this range. This multi-step negotiation approach allows the agent optimize a negotiation result in a given range, however, it does take longer and involves greater negotiation cost. Another example of multi-step negotiation is as follows: the agent first proposes that 3 units of MQ be transferred; if the other agent accepts it, the first agent saves 2 units of MQ; otherwise, it can raise the amount of MQ units transferred to 4 or 5.

For a multi-linked negotiation problem, the search algorithm with the support of the partial order scheduler provides a value assignment for each feature under negotiation. The agent can perform a single-step negotiation using this feature assignment as guide, and find the negotiation result quickly. Alternatively, the agent can perform a multi-step negotiation within the limit provided by the value assignment in the negotiation approach. In the latter approach, the agent may find a slightly better solution, but with the extra negotiation cost in terms of time and other resources.

5.7.3 Difficulty Measure of the Negotiation Problem

In Chapter 5, we presented a measure for the difficulty of the negotiation problem, which is based on the number of constraints in the agents' local task plans. We have examined a number of other measures as well, but we have not found a perfect predictor of the difficulty of the negotiation problem for the following reasons:

1. Since the negotiation problem we studied is actually distributed as two sub-problems, it is not clear how to combine the characteristics of the sub-problems to describe the whole problem.

2. The solution space is not continuous. Because the DTC scheduler does not support the task-breakable assumption, the task has to be fitted into a single time slot completely.

However, from the experimental work, we did find that the initial range of the first proposal is a very important factor in predicting how difficult the negotiation would be (See Section 5.6.4). This idea is used in Chapter 2.5.1, where the success probability depends on the flexibility. In Chapter 2, the measure of the difficulty of the negotiation problem is described as a success probability function. The success probability of a negotiation depends on the parameters of the function: several features of the negotiation issue, such as the flexibility, the transferred reward, the negotiation start time, etc. These features are decided by the contractee agent. On the other hand, the success probability also depends on the function itself. The structure of the function depends on the contractor agent's problem solving context and its decision making process. How to build such a function is still an open question.

In this section, we discussed the different negotiation approaches which have been presented in this chapter and previous chapters. We compared the functionality of the DTC scheduler and the partial order scheduler, and how they affect the negotiation methods. We also have analyzed the implications of the single-step negotiation and multi-step negotiation mechanisms. Two different approaches to measure the difficulty of a negotiation problem have been summarized here too.

5.8 Summary

In this chapter, we present a cooperative negotiation mechanism in which negotiating occurs over a multi-dimensional utility function. We show the application of this mechanism to the task allocation domain in a cooperative system. The contractee agent has a task

that needs to be performed by other agents. To perform this task, the contractor agent can choose from several alternatives that produce different qualities and consume different resources. This context requires a complex negotiation that leads to a satisfying solution with increasing combined utility. We started with a binary search algorithm as a mechanism to find a compromise between the contractee's protocol and the contractor's counter-proposal. After examining the trace of this negotiation mechanism carefully, we developed a better way to do the distributed search explicitly in the agents' negotiation. The range-by-range algorithm searches a broader space and exploits the domain knowledge from the previous communication to improve the negotiation process. Instead of a tightly constrained proposal, the range proposal allows the contractor agent to have more freedom to react, which improves the efficiency of the negotiation. The multi-step negotiation mechanism is actually an anytime mechanism: by investing more time, the agent may find a better solution. A set of multi-step protocols are developed based on this mechanism. Experimental work is done to study how different protocols work in different situations. A complete search is performed as a baseline for comparison. We find a phase transition phenomenon: When the negotiation situation is very simple or very difficult, the extra negotiation effort does not bring reasonable extra gain. When the negotiation situation is of medium difficulty, the extra gain exceeds the extra effort. The meta level information could be used to provide advice on how the agent should choose the protocol to balance its gain and effort of negotiation. We develop several predictors to measure how difficult the negotiation could be to help the agent to choose the appropriate protocol. This is a very hard problem. Although we have not found a perfect predictor, we did find some simple measures that are helpful.

We would like to continue this work to obtain a better understanding of the negotiation problem characteristics. These characteristics should help us rate negotiation problems by their difficulty, estimate the probability of finding a good solution before the negotiation is even started, and ultimately, help the agent make a more reasonable decision about the probable cost and duration of a negotiation, and potential gain.

CHAPTER 6

SELECTED WORK ON NEGOTIATION

In this chapter, we will look at some other areas of research on negotiation and how they relate to the work in this thesis. The following topics are grouped by their approaches to the negotiation problem.

6.1 Contract Net Protocol and Bounded Rational Self-Interest(BRSI) Negotiation

The earliest work on negotiation can be traced back to Smith's work[60] on the contract net protocol (CNP), which provides the basic model of the negotiation process used in this thesis. The CNP is modeled on the contracting mechanism used by businesses to govern the exchange of goods and services. It provides a solution for finding an appropriate agent to work on a given task based on the evaluation of the bids from agents who can perform that task. This work established a basic negotiation protocol model that involves a process of task announcing, bidding evaluation, and contract awarding. It allows a mutual selection by both managers and contractors. However, work on the CNP discusses no formal model for making decisions about task announcing, bidding and contract awarding.

TRACONET[51] extends the CNP work by introducing a formalization of the bidding and awarding decision process. This formalization uses marginal cost calculations based on local agent criteria. The pricing mechanism generalizes the CNP to work for both cooperative and competitive agents. In addition, TRACONET extends the CNP framework to handle task interactions by clustering tasks into sets to be negotiated over as atomic bargaining items. Since an agent can generate bids for multiple contracts simultaneously, how the agent sets the bid price depends on the agent's assumptions about which contracts the agent will actually receive. In order to examine how an agent's method for pricing a bid influences the agent's utility, work on TRACONET compares an opportunistic pricing policy to a safe pricing policy. It finds that opportunism speeds up the negotiations, but safe policies guarantee monotonic decrease of the local cost. While the TRACONET work extends the basic CNP by introducing a decision making model and utility function calculation, its negotiation process does not allow counter proposals and focuses on a single issue only (the price).

Significant work[47] has been done to extend the above work based on the use of bounded rational self-interest agents (BRSI). This research concerns BRSI agents that deal intelligently with the uncertainty present in the negotiation process. The uncertainty comes from three directions: uncertainty as to future events, uncertainty as to the results of concurrent ongoing negotiation events, and uncertainty in the approximate computation of utility.

The first aspect of this work concerns the negotiation protocol about the commitment formation process. Leveled commitments [47, 48] allow an agent to decommit by paying a decommitment penalty. Different leveled commitment protocols (Original contract [one task to move at a time]; Cluster contract [move two or more tasks between two agents]; Swap contract [two agents swap tasks]; Multi-Agent contract [at least three tasks to be transferred between at least three agents]) and their parameterizations are empirically compared to each other and to several full commitment protocols[1, 2]. Concerning solution quality, the leveled commitment protocols are significantly better than the full commitment protocols of the same type, but the difference between the various leveled commitment protocols are minor. The second aspect of this work concerns the tradeoffs between negotiation risks and computation costs[3, 50]. The implication of the deadline and timeouts on negotiation is also studied. In our work, the agents are assumed to be bounded rational but not necessarily to be self-interested. The techniques dealing with uncertainty in the above work are potentially helpful. For example, the idea of a decommitment penalty fits our work.

Hunsberger[25] and Sandholm[49] have studied combinational auctions, in which there are multiple items for sale, participants who may place bids on arbitrary subsets of those items, and an auctioneer who must determine which awardable combination of bids maximizes revenue. It allows agents to select a shared plan for the group through a distributed computation process. This work relates to our work in that the agent has multiple issues to negotiate over with multiple agents and tries to find a best combinational solution. The difference is that there is no second round proposal in the auction, and the bids are not changeable after they are sent out. Also the selection of an awardable combination is a centralized process performed by one agent.

Collins, etc.[26, 14] introduce the concept of flexibility to the contracting mechanism for bid evaluation. This is the only work we found related to flexibility in negotiation. The bid evaluation process in this work incorporates costs, task coverage, temporal flexibility, plan feasibility, and risk estimation. The contractor agent takes into account the flexibility of the commitment when it calculates the bid price. However, there is no explicit reasoning about how to balance the flexibility and the feasibility as we do in this thesis.

Although multiple simultaneous negotiations are allowed in the above work, the agent does not reason explicitly about the relationship among concurrent negotiation issues. A statistical model is used by the agent to predict future events when it evaluates the current commitment. Also, the relationship between two negotiation issues is simply the competition between the agent's local resource (*indirectly-linked*); no negotiation chain that spans more than three agents is taken into account here.

6.2 Negotiation in Cooperative Distributed Problem Solving (CDPS)

In the cooperative distributed problem solving (CDPS) area (i.e. DENEGOT[36]), negotiation is viewed as a distributed search through potential compromises. To estimate the quality of potential solutions, the negotiation search space is structured into a lattice of sets of potential compromise solutions based on hard constraints. A solution in a higher set in the lattice, if it is achievable, will be preferable to a solution in a lower set. Agents first

search under the hard constraint level representing the highest quality solution standard achievable in the current situation. By relaxing hard constraints, the set of compromises that qualify as solutions is enlarged. Agents search for a resolution under the relaxed hard constraint set when a solution can not be found under the current set of constraints. The limitation is the assumption that individual agents are only involved in one negotiation session at a time; a negotiation session must end before another agent can become involved.

Similarly, Lander and Lesser[30] report on an approach called *negotiation search*: an algorithm that explicitly recognizes and exploits conflict to direct search activity across a set of agents. A set of states is defined which represent the states of the negotiation process with respect to the quality of the actual solution. The states are changed by applying negotiation operators to initiate, critique, extend, and relax solutions, and to terminate the search. Laasri et. al. describe a generic model, the recursive negotiation model (RNN) of multi-agent problem solving that details various situations that can potentially benefit from negotiation [29]. This model defines where and how negotiation can be applied during problem solving based on structuring problem-solving into four stages: problem formulation, focus-of-attention, allocation of goals or tasks to agents, and achievement of goals or tasks. For the Belief-Desire-Intention(BDI) agent framework, a collaborative negotiation process [12, 13] is used to resolve the conflicts in the shared plan of actions and the shared beliefs. It is a recursive *Propose-Evaluate-Modify* cycle of actions through which agents exchange evidence and justification to adapt their beliefs. In these approaches, negotiation is used to solve conflict in a cooperative problem-solving environment; agents need to work synchronously on their problem-solving negotiation processes. No task or resource is transferred among agents through negotiation, and agents don't need to explicitly reason about the utility of alternative proposals/counter-proposals.

A multistage negotiation paradigm is used for solving distributed constraint satisfaction problems[15]. Agents first make tentative commitments to local alternatives that would serve to partially satisfy the goals for which they are responsible. In making a tentative commitment, an agent removes the affected resources from the pool of available resources. After making a tentative commitment, an agent requests that other agents confirm this commitment by making their own local commitments in ways that are compatible. If some agents can not do so, an iterative exchange results in the construction of induced exclusion sets and goal exclusion sets in the agents. The process of attempting satisfaction of multiple global goals occurs simultaneously in the network. It is sometimes necessary to retry alternatives that had previously failed when tentative commitments have been retracted. This idea is extended and applied to the distributed meeting scheduling problem.

In the distributed meeting scheduling process [57, 58, 59], a host agent announces the meeting time with one or more possible options. Other attendant agents look at their schedules to find a "yes/no" answer or to provide alternatives. The commitment strategies could be committed (the agent tentatively blocks the time it proposed) or non-committed (the agent does not block the time until an agreement is reached). Multiple meeting scheduling processes could be going on concurrently, and the negotiation process on one meeting would affect other meetings. Hence, the commitment strategy is an important factor which affects the system's performance. Adaptive choice of commitment is recommended according to the combination of environmental factors. The choice of search biases in scheduling meetings also affects the frequency of conflicts between new processes and completed

meeting scheduling processes. The problem domain studied here is a relatively narrow domain and the relationship among multiple negotiation issues is the competition for local resources (time). The policy to reduce conflicts is to choose an appropriate strategy from a statistical view point, and involves no explicit reasoning about particular related issues when building or evaluating proposals.

Progressive negotiation [33, 32] is proposed to deal with negotiation that involves multiple homogeneous agents. The negotiation among a group of agents can be divided into a number of sub-negotiations which proceed incrementally. The agents are cooperative and can always negotiate to increase their mutual benefits. This approach is not suitable for our problem because we have a more general problem domain and more complex agent systems.

The above work focuses on the research into negotiation process models. All agents are assumed to be cooperative and working on a common goal. Negotiation is mainly used for conflict resolution.

6.3 Game Theory in Negotiation

Game theory provides a formal framework for negotiation. However, it frequently makes simplifying assumptions to facilitate the mathematical analysis. This approach includes building a formal model of negotiation, developing appropriate negotiation protocols and strategies and analyzing their equilibrium. Traditionally, game theory can be divided into two branches: cooperative and non-cooperative game theory. Cooperative game theory abstracts away from specific rules of a game and is mainly concerned with finding a solution given a set of possible outcomes. A topic like coalition formation is typically analyzed using cooperative game theory. In such games, a surplus is created when two or more players cooperate and form a coalition. Cooperative game theory can then determine how the surplus is to be divided given a coalition and a set of assumptions. Non-cooperative game theory, on the other hand, is concerned with specific games with a well-defined set of rules and game strategies. All strategies and rules are known beforehand by the players. The notion of equilibrium strategies is to determine "rational" outcomes of a game.

Zlotkin and Rosenschein import game theoretic techniques into multi-agent negotiation ([76, 75, 78, 77] [44]). They studied the task-oriented domain, the state-oriented domain and the worth-oriented domain and explored the relationship between certain domain characteristics and the negotiation mechanism.

Based on this game theoretic approach, Kraus, Schwartz and Wilkenfeld developed a strategic model of negotiation that takes the passage of time during a negotiation process itself into consideration [53, 6]. Changes in the agents' preferences over time will change their strategies in the negotiation and, as a result, the agreements they are willing to reach. This model is applied to the data allocation problem in MultiAgent environment.

However, several assumptions limit the practical applicability of game-theoretic results and distinguish this work from ours. Common assumptions are, for instance: (1) complete knowledge of the circumstances in which the game is played (the rules of the game and the preferences and beliefs of the players are "common knowledge") (2) full rationality of the

players. *Isolated Negotiation* is also assumed in this work; each negotiation stands alone with no relationship between any two negotiation issues.

6.4 Behavioral Science in Negotiation

Behavioral and social sciences study human cooperation and coordination and develop frameworks and models of organizations and communities. Their theory and knowledge are beneficial for research in negotiation. Sycara presents the PERSUADER system [61] that solves conflicts through direct negotiation among the interacting agents or through a third party, the mediator. It is based on case-based reasoning and multi-attributed utility theory. Argumentation [45] is introduced as negotiation mechanism for agents with the BDI (belief-desire-intentions) architecture. Since it is an iterative process that emerges from exchanges among agents to persuade each other and bring about a change in intentions, it can be used for achieving cooperation and agreement. The above work introduces the idea that the agents can change their criteria functions through information exchange during the negotiation process, when they find other criteria functions that are more suitable for the current situation. This idea is related to the "relaxing constraints" work we plan to do (See Section 7.3).

Sathi's work [54] on resource reallocation problems adopts the approach through constraint directed negotiation. The definition of the problem includes resources, requests for resources and a set of constraints. The solution contains buy and sell bids to indicate resource transactions from one agent to another. The problem is solved using a search technique. Agents search individually in the early stages and as a group in the later stages. A number of qualitative relaxation operators were developed based on human negotiation problem solving, such as: bridging, reconfiguration and logrolling. These operators are used to relax constraints and generate new alternatives. The search uses several aspects of constraints, such as constraint importance, looseness, utility and threshold levels. This work relates to our work on cooperative negotiation where agents perform a distributed search process. However, in this work only qualitative reasoning is performed which is used to identify alternatives but not to evaluate how good an alternative is. The constraint evaluation and relaxation techniques may be helpful in our future work of "relaxing constraints" (Section 7.3).

Faratin, Sierra, and Jennings [19] present a formal model of an autonomous agent's decision function as it relates to the process of service-oriented negotiation. The negotiation is about many issues between two parties. The model defines a number of tactics (time-dependent, resource-dependent, and behavior dependent) which the agent can employ during negotiation to generate offers and counter-offers, and evaluate proposals and offer counter-proposals. Also, this model indicates how an agent can change these tactics over time given strategic behavior. This model of negotiation is applied in the business process management system ADEPT [38, 39]. The idea that the agent dynamically chooses negotiation tactics is similar to our approach in which the agent chooses a negotiation strategy in real-time based on its relationships with other agents and the issue under negotiation.

6.5 Learning in Negotiation

Several learning technologies are added to negotiation to improve negotiation performance. When agents have incomplete information about each another, it becomes important to learn about another agent by observing its behavior during negotiation. A simple learning method based on Bayesian classifiers allows an agent to make predictions about another agents' preferences by building statistical models of other agents' preference functions from its past interactions with them [7]. The learning mechanism helps to reduce the amount of communication needed, and thus improve the overall efficiency of the negotiation process. Similarly, Bayesian beliefs are used to learn about other agent's reservation price [71]; these beliefs are updated depending on the opponent's moves. However, once both agents use beliefs to determine their strategies, they also need beliefs about their opponent's beliefs, and so on. This is known as the problem of "outguessing regress". In practical applications, this problem is circumvented by assuming bounded rationality.

The generic program approach is the basic technique used to make negotiation systems adaptive [35, 40]. Oliver [40] demonstrates a system of adaptive agents that can learn effective negotiation strategies. Computer simulations of "alternative-offers" negotiations are presented. Binary coded strings represent the agents' strategies. Two parameters are encoded for each negotiation round: a threshold which determines whether an offer should be accepted or not and a counter offer in case the opponent's offer is rejected. These elementary strategies are then updated in successive generations by a genetic algorithm.

Reinforcement learning technology has been used for agents to learn to choose the appropriate coordination mechanism [18]. Excelente-Toledo and Jennings have analyzed the use and the efficacy of agents learning about making decisions about when and how to coordinate. They showed that learning improves the decision making when agents are uncertain about the other agents' actions. They also showed that learning was ineffective they can make accurate predictions about their environment and other agents. One major difference of this work and our work is that the environment their agents work in is much simpler than our agents'. Grid-world scenario is used as the test bed in their work, hence the agents only deal with simple tasks, and one task at a time.

The above learning technique can also be applied to our work to enable agents to learn the relationships between each other and, hence to be able to select an appropriate strategy (See Section 7.3).

CHAPTER 7

CONCLUSION AND FUTURE WORK

To negotiate efficiently and dynamically in a complex multi-agent system, an agent needs to analyze its current set of negotiation issues, choose the appropriate negotiation approaches, and balance the gain and cost of doing each negotiation. In this thesis work, we have studied these problems from three different negotiation research perspectives: decision-making in negotiation, negotiation process and negotiation language.

In this chapter, we will review the major ideas and conclusions from experiments that we have presented previously in this thesis. We will then return to the research issues introduced in Chapter 1, and discuss how our approaches address these issues. Next we will summarize the contributions of this thesis, including both intellectual contributions and technical contributions. Finally, we will point out the remaining questions which are open for future research.

7.1 Research Issues Revisited

In Chapter 1, we introduced the motivation for this thesis: to explore a number problems concerned with sophisticated negotiation in multi-agent systems. These problems include multi-linked negotiation, negotiation in a complex organizational context and cooperative, multi-step negotiation. Now we will look at how the approaches we have developed, implemented and evaluated address these problems.

7.1.1 Multi-Linked Negotiation

The multi-linked negotiation problem is a new problem which has not been recognized before this work. It describes a situation where an agent needs to deal with multiple negotiation issues which are related to each other. In Chapter 2, this problem is introduced through a supply chain example. The relationship among these related negotiation issues are analyzed and classified into two categories: the *directly related* relationship and the *indirectly related* relationship. In order to minimize the conflicts and maximize the utility in a multi-linked negotiation problem, the agent needs to find an appropriate partial ordering of these negotiation issues and assign suitable values for those attributes under negotiation. First, a partial order scheduler is constructed which enables the agent to reason about the time related constraints on different negotiation issues. This reasoning tool enables the agent to recognize situations where it is possible to concurrently negotiate individually on different issues without worrying about interactions among their solutions. The partial order scheduler also enables the agent to reason about and manage the flexibility

attached to each negotiation issue. Experimental work shows that effective reasoning about and management of the flexibility in multi-linked negotiation can significantly improve the performance.

Furthermore, a formalized model based on the relationships among negotiation issues is presented for the multi-linked negotiation problem. Each negotiation issue is represented as a node with links to other issues representing its directly-related relationship with those other issues. Each negotiation issue has a set of features (attributes) associated with it. These features either have been decided by other agents or the environmental circumstance or need to be negotiated over. The *success probability* of an issue describes how likely the negotiation over this issue will be successful. It depends not only on the start time of the negotiation, which is decided by the ordering of all the negotiation issues; but also on other features of the issue, such as the reward and the flexibility. Based on this model, a search algorithm can be used to find the best negotiation approach, including how to order the set of negotiations and what values are assigned to those features under the negotiation. A best negotiation approach means that if the agent follows this approach, its expected utility will be maximized. We developed a heuristic search algorithm, which significantly reduces the search effort. Experimental work has evaluated the negotiation approaches generated by this search algorithm and performance is greatly improved over those simple pre-defined approaches.

7.1.2 Integrative Negotiation

In a complex agent society, an agent will need to work with other agents from a variety of different organizational positions; hence the agent needs to quantitatively reason about each negotiation session so as to choose an appropriate negotiation strategy. In Chapter 3, we presented an integrative negotiation mechanism, which allows the agents to choose a negotiation strategy along the spectrum from one that is purely self-interested to one that is completely cooperative. This mechanism is based on the MQ framework previously developed by Wagner [67, 66]. The *relational MQ* is introduced into this framework as a way of supporting a range of negotiation strategies, and an agent's preference for *relational MQ* represents how cooperative it is with other agents concerning certain negotiation issues. Different preference functions, including both linear functions and non-linear functions, can be used to represent relationships between agents. This uniformed mechanism allows the agent to choose different negotiation attitudes using the same negotiation protocol. The negotiation attitude can be easily adjusted according to the agent's organizational goals and the current environmental circumstance.

An example based on the supply chain scenario is used to demonstrate how this mechanism works. Experimental work shows how the agent's attitude toward other agents affects the agent's performance and the social welfare. It is found that it may not be a good idea to always be *completely cooperative* in a situation involving an unknown agent's assistance; in that case, choosing to be partially-cooperative may be good for both the individual agent and also for the society.

7.1.3 Multi-Leveled Negotiation

Given multiple related negotiation issues, the complex organization context and the uncertainties of the execution characteristics, negotiation should be performed at different abstraction levels. In Chapter 4, a multi-leveled negotiation framework is presented. This framework is not only a negotiation framework, but is a complete agent architecture. It provides the agent with planing, scheduling, negotiating and executing functionalities. The following technologies have been integrated into this architecture: the MQ framework and MQ scheduler which support reasoning about organizational concerns and utility, the TÆMS framework and the DTC scheduler which handle detailed feasibility analysis and implementation of objectives, the multi-linked negotiation reasoning technology and the partial order scheduler, the integrative negotiation mechanism, and technique for lower level renegotiation and execution monitoring. With this architecture, the agent can benefit from the proper integration of these technologies, and handle the sophisticated negotiation problems discussed above.

Two different reward models are introduced which allow the agent to explicitly reason and negotiate over flexibility. A measure of uncertainty is developed so that the agent can dynamically choose a proper reward model according to its current problem solving context. The experimental work shows how this two-leveled negotiation mechanism provides the agent with a better way to handle uncertainty of task duration, and the ability to dynamically choose the reward model for negotiation improves the agent's performance.

7.1.4 Cooperative Negotiation

In Chapter 5, we look at task allocation through negotiation among cooperative agents. A multi-step, multi-dimensional negotiation mechanism is developed. This mechanism uses marginal utility gain and marginal utility cost to structure this search process, so as to find a solution that maximizes the combined utility. These two utility values together with temporal constraints summarize the agents' local information and reduce the communication load. Since the goal is to find a solution which increases the combined utility of the negotiating agents, the negotiation can be viewed as a distributed search over the possible solution space. Two search algorithms, a "range-by-range" search and a binary search, are developed to guide the search in a multi-dimensional space.

A set of protocols are constructed based on this mechanism, which involve different levels of the negotiation effort. By investing more effort on negotiation, the agents increase the likelihood of getting a better solution. A measure of the complexity of the negotiation problem is developed based on the number of constraints in the agents' local task plans. The experimental result shows a phase transition phenomenon as the complexity of negotiation situation changes: when the negotiation situation is very simple, a simple single-step negotiation brings a fairly good solution; when the negotiation situation is very difficult, extra effort does not bring reasonable extra gain; when the negotiation situation is of medium difficulty, then the extra gain exceeds the extra effort. Furthermore, we show it is possible for the agent to choose an appropriate negotiation protocol according to the complexity of the negotiation problem, in order to balance the gain and the cost of the negotiation.

7.2 Contributions

The contributions of this thesis work can be described from two perspectives: the intellectual contributions and the technical contributions.

7.2.1 Intellectual Contributions

As the technology of multi-agent systems is used for increasely complicated applications, the interaction among intelligent agents becomes increasely important. Negotiation, a technique used for task allocation, resource allocation, and conflict resolution becomes an important area for study in both multi-agent systems and intelligent agents.

There is one major difference between this work and other work on negotiation. Negotiation, here, is not viewed as a stand-alone process, but as one part of the agent's activity which is tightly interleaved with the planning, scheduling and executing of the agent's activities, which also may relate to other negotiations. Based on this recognition, this negotiation research involves concerns more complicated than the protocol and the language.

In addition to studying the isolated, single negotiation issue, we study the multi-linked negotiation problem for the first time. Based on our work, agents are now able to handle the multiple, complex tasks that are involved with multiple negotiation issues more efficiently. It opens a research area for agent's planning and decision-making on multiple interrelated issues; the results of those issues are uncertain and partially depend on the agent's decision process but also depend on the environment and the other agents' problem solving contexts.

The introduction of integrative negotiation bridges the gap between self-interested negotiation and completely cooperative negotiation. It allows negotiation to become a suitable technique not only in marketing systems or cooperative systems, but also in those multi-agent systems with complex organizational relationships. Additionally, this integrative negotiation mechanism built on the MQ framework provides a test bed for future study of how organizational structure affects the performance of the individual agents and the overall system. It also opens a door for computational simulation of human society and organizations.

The work on multi-leveled negotiation allows negotiation to be performed at different abstraction levels. The negotiation is enriched by introducing dynamic commitments - commitments which can be adjusted later according to pre-defined agreements. Flexibility is connected to uncertainty in an agent's reasoning process, which enables the negotiation to be fitted into applications with real-time and dynamic characteristics.

This multi-level agent architecture integrates the MQ framework and the TÆMS framework seamlessly. It provides a framework in which the agent can reason about the organizational concerns, implementation of objectives, and negotiation decisions. There are potentially future studies about intelligent agents, which deal with multiple, sophisticated tasks and are involved in complex organizational contexts, that can be performed based on this architecture.

Furthermore, the cost of negotiation is explicitly reasoned about, compared with the gain from the negotiation. The complexity of the negotiation problem is measured and used to guide the agent to select an appropriate negotiation protocol, so that it can balance the gain and the cost from the negotiation. The *success probability* is another predictor of

the difficulty of the negotiation problem. These measures are domain independent and can be used for other applications too.

The topic of negotiation is extended to include multiple attributes: the time scope for the task, the requirement of quality of achievement, the transferred MQ, the flexibility and the reward model. Multi-dimensional negotiation enlarges the solution space and makes it more likely to find a good solution.

In summary, this thesis addresses many important issues that will arise as negotiation is used in the next generation of more complex multi-agent applications.

7.2.2 Technical Contributions

The technical contributions of this thesis work can be summarized as the following:

- Implementation of the two-level agent architecture, including: the MQ Agent Controller, the Agent Negotiator, the Agent Executor, the interface with the MQ scheduler, and the interface between the MQ tasks and TÆMS tasks.

- Development of the multi-leveled negotiation mechanism and introduction of a measure for uncertainty, which enables the agent to dynamically choose the reward model and explicitly reason about the flexibility and uncertainty.

- Experimental work that shows that the two-level negotiation mechanism enhances the agent's ability to cope with uncertainties, and that the measure of uncertainty allows the agent to make a decision about how to balance the flexibility and cost in negotiation, and hence improves the agent's performance.

- Development of the partial order scheduler and a set of related reasoning algorithms for use in reasoning about the time-related constraints and flexibilities of agent's activities. Experimental work shows that the reasoning about and management of flexibility in multi-linked negotiation problems improves the agent's performance.

- Proof of the validity of the algorithm which produces a linear schedule from a partial order schedule. The algorithm is built with the "breakable" assumption; the execution of a task can be interrupted and resumed later. This assumption allows the agent to exploit its process time more efficiently.

- Development of a formalized model for multi-linked negotiation problems, which allows agents to reason about multiple inter-related negotiation issues and evaluate a negotiation approach that specifies the ordering and the parameterization of negotiations.

- Implementation of two search algorithms for multi-linked negotiation problems. The complete search algorithm finds the best negotiation approach, while the heuristic search algorithm finds a good negotiation approach with limited search effort.

- Experimental work that demonstrates that a reasonable negotiation approach to multi-linked negotiation problems decreases the likelihood of the need for decommitment from previously settled issues and increases the likelihood of utility gain.

- Extension of the MQ framework by introducing *relational MQ* for use in representing varying negotiation strategies from self-interesting to completely cooperative.
- Experimental works that shows how the agents' choices of different negotiation strategies affect the agents' performance as well as the social welfare under a specific situation, which also provides information for the agent to choose an appropriate negotiation strategy in this specific situation.
- Development of a multi-step, multi-dimensional cooperative negotiation mechanism, with two different search algorithms that guide the multi-dimensional negotiation.
- Development of a complexity measure of the negotiation problem, and the analysis of a number of other possible measures.
- Experimental results that show a phase transition phenomenon as the complexity of negotiation situation changes.

7.3 Future Research Directions

We conclude by documenting several directions for future research.

1. *The use of meta-level information.* In Section 2.4, we have defined the *success probability* to describe how likely the negotiation is to succeed. It is a function that depends on a set of features. However, it is not clear how to construct such a function to model the difficulty of the negotiation. One possible approach is for the agents to communicate meta level information before negotiation, such as the slack time in the agent's schedule, the number of other competitors, etc. This information could be used by the agent to construct the function more accurately. The communication and reasoning process of meta-level information can be extended to a chain which covers more than two agents. For example, agent A can reason not only about the meta level information from agent B but also about the information from agent C, and the combination of them, given the knowledge that agent B also negotiates with agent C at the same time as it negotiates with agent A.
2. *Learning to build the model of negotiation outcomes.* The function of *success probability* is to describe how the likelihood of a successful negotiation depends on a set of features in the negotiation. It is possible for an agent to learn to construct and adjust the structure of this function based on its previous negotiation experience, if the agent has similar negotiation experience in the past. A questions arises here: When agent A is learning about its negotiation experience with agent B, agent B is also learning about its negotiation experience with other agents; if the negotiations between agent B and other agents are related to the negotiation between agent B and agent A, the learning result of agent B will affect its decision on the negotiation with agent A and hence affect agent A's learning result. Thus this presents an interesting multi-agent learning problem.

3. *Negotiation approach as a policy.* In the model of the multi-linked negotiation problem (Chapter 2), only two possible negotiation outcomes are considered: "success" or "fail". "Success" means a mutually agreed commitment is formed given a set of pre-defined attributes; otherwise, the negotiation is considered to "fail". Actually, when negotiation is successful, there are potentially many different outcomes depending on the parameters in the commitment. For example, when agent A asks whether agent B can perform task t before time 30, if agent B accepts this proposal, it can promise to finish task t at a time earlier than 30 to earn an extra reward. Depending upon the promised finish time of task t, agent A can adjust its negotiation on other tasks which are related to task t. Different promised finish times of task t represent different negotiation outcomes. These additional outcomes can be grouped into a number of ranges, such as: "the promised finish time falls into $[0, 10]$", "the promised finish time falls into $[10, 20]$", and "the promised finish time falls into $[20, 30]$". The negotiation process can be modeled as a Markov decision process, and the negotiation approach can be generated as a policy: perform the negotiation according to the results of the previous negotiations.

4. *Learning to establish long-term relationships.* An integrative negotiation mechanism is presented in Chapter 3, which allows the agent to choose any degree of cooperation in negotiation. How cooperative should an agent be toward another agent concerning a special negotiation issue? This also opens another interesting multi-agent learning problem. The agent can learn from its previous interactions with the other agent, and also learn from the results of past cooperation strategies. So the agent can dynamically adjust its negotiation strategies.

5. *Relaxing constraints for negotiation.* There are a number of constraints affecting the difficulty of the negotiation problem and the potential outcome of the negotiation (See Section 5.5). There are both "hard" constraints and "soft" constraints. "Hard" constraints can not be changed; "soft" constraints can be changed or removed. When an agent recognizes that the current negotiation situation is too difficult, it can reason about how to remove or change some constraints to make the negotiation easier. The relaxing of constraints may cause a decrease of local utility; however, it also makes negotiation more likely to succeed. Hence the agent needs to reason about which constraints to relax and how to relax them.

In summary, this thesis addresses a number of issues in sophisticated negotiation, which are important for intelligent agents and multi-agent applications. It provides approaches to these problems in either heuristic or formal ways. It points out several open questions for future study.

APPENDIX A

IMPLEMENTATION INFORMATION

The agent architecture and all the negotiation mechanisms and algorithms described in this thesis, except where noted, have been implemented in the Java language. There are roughly 31,000 lines of code, including comments. Rodion Podorozhny implemented part of the cooperative negotiation mechanism described in Chapter 5. Sherief Abdallah implemented the search algorithm described in Section 2.4.3.

The experiments are performed in the MASS simulator environment[22], and the agents were built using the JAF agent framework[62].

APPENDIX B

REASONING ALGORITHMS IN CHAPTER 2

Algorithm 2.2.1 *Propagate_EST_DL*

Given a set of tasks with the outside constraints of the earliest start times and deadlines, the durations and the precedence relationships, this procedure finds t.est and t.dl for each task t according to the definition in Section 2.2.1.

Input: a set of tasks V, precedence relationship represented as E, outside constraints associated with tasks.

begin

update the EST:

```
S = V.clone();
while S is not empty
      find a task t in S which has no pretask in S
         t.est = max ( eft[Pre(t)], t.est_o );
         t.eft = t.est + t.process_time;
remove t from S;
if can not find such a task t and S is not empty, report error: there is a cycle in the graph.
```

update the DL:

```
S = V.clone();
while S is not empty
      find a task t in S which has no posttask in S
         t.dl = min ( lst[Post(t)], t.dl_o );
         t.lst = t.dl - t.process_time;
      remove t from S;
if can not find such a task t and S is not empty, report error: there is a cycle in the graph.
```

end

Algorithm 2.2.2 *Get_Earliest_Finish_Time*

Calculate the earliest finish time eft[V] of a set of tasks V.

Input: a set of tasks V. Each task t has earliest start time t.est and duration t.process_time

begin

```
sorting the tasks in V according to their earliest start time in increasing order;
eft = min_value;
max_eft = min_value;
for each task t in V
      if (t is local)
         if (eft >= t.est) // there is no gap
            eft = eft + t.process_time;
```

```
            else // there is gap
                eft = t.est + t.process_time;
        else // t is a nonlocal task
            if(max_eft <t.finish_time [1])
                max_eft = t.finish_time;
    return max(eft, max_eft);
end
```

Algorithm 2.2.3 *Get_Latest_Start_Time*

Calculate the latest start time of a set tasks V: $lst[V]$.
Input: a set of tasks V. Each task t has deadline t.dl and duration t.process_time

```
begin
    sorting the tasks in V according to their deadlines in decreasing order;
    lst = max_value;
    min_lst = max_value;
    for each task t in V
        if (t is local)
            if ( lst <= t.dl) // there is no gap
                lst = lst - t.process_time;
            else // there is gap
                lst = t.dl - t.process_time;
        else // t is a nonlocal task
            if (min_lst > t.start_time)[2]
                min_lst > t.t.start_time;
    return min(eft, min_lst);
end
```

Algorithm 2.2.4 *Feasible_Schedule*

Translate a partial order schedule to an executable feasible linear schedule if the partial order schedule is valid, otherwise report failure.
Input: a partial order schedule (V, E) after the processing of the Propagate_EST_DL procedure. Each task has its est and dl with respect to its pretasks, posttasks and its outside constraints.
Output: a feasible linear schedule if there exists one; otherwise report failure.
start_time: the start time of the schedule

```
begin
    sorting the tasks in V according to their earliest start times in increasing order;
    st_i = max( start_time, est(V[0]));
    st_ii = Get_Next_Check_Point(V, st_i)[3];
```

[1]For a nonlocal task, the finish time should be the promised finish time in the commitment. Before the commitment is available, the agent can assume $t.finish\_time = t.est + t.process\_time$ for the initial reasoning process. It can be another number based on other knowledge available to the agent at this time.

[2]For a nonlocal task, the start time should be the earliest start time in the commitment. Before the commitment is available, the agent could assume t.start_time=t.dl + t.process_time for the initial reasoning process. Also it could be any time based on that assumption.

[3]See description in below.

```
sortingByDL(V, st_i)[4];
while V is not empty
      in_process = false; // if still in progress
      task_ready = false; // if there is a task ready
      for every task t in V
         if (t is ready)
            task_ready = true;
         if (t is nonlocal && t.finish_time<=st_i)
            remove t from V;
            update ready flag for all tasks in V;
            in_process = true;
      end of for
      for every task t in V
         if (t is ready && t is local && t.est <= st_i)
            if( st_i + t.need_process_time < t.dl )
               print ("t is late!");
               return null;
            if( st_ii - st_i < t.need_process_time)
               schedule.add ((st_i, st_ii, t.label);
               t.need_process_time = t.need_process_time - (st_ii - st_i);
               st_i = st_ii;
               st_ii = Get_Next_Check_Point(tasks, st_i);
               in_process = true;
            if( st_ii - st_i >= t.need_process_time)
               schedule.add (st_i, st_i+t.need_process_time, t.label);
               t.need_process_time = 0;
               remove t from V;
               st_i = st_i+t.need_process_time;
               st_ii = Get_Next_Check_Point(tasks, st_i);
               in_process = true;
            sortingByDL(V, st_i);
            update ready flag for all tasks in V
            break;
      end of for
      if(task_ready && !in_progress)
         schedule.add ((st_i, st_ii, slack_time);
         st_i = st_ii;
         st_ii = Get_Next_Check_Point(tasks, st_i);
         sortingByDL(V, st_i);
      else if(!task_ready)
         print("no progress for schedule!");
         return null;
end of while;
```

[4]Sorting all tasks in V that start at or before st_i according to their *deadlines* in increasing order.

```
    return schedule;
end
```

Get_Next_Check_Point(V, st) Find the next time point for that should be checked.

```
begin
    min_dl = max_value;
    for every task t in V
        if (t.est <= st)
            if (t.dl < min_dl && t.dl > st)
                min_dl = t.dl;
        else
            return (MIN(min_dl, t.est ));
    return min_dl;
end
```

Algorithm 2.2.5 *Range_Evaluation*

Determine if a partial order schedule is valid.
Input: a partial order schedule (V, E) after the processing of the propagate_EST_DL procedure. Each task has its est and dl with respect to its pretasks, posttasks and its outside constraints.
Output: return true if this partial order schedule is valid; otherwise return false.
slot [est, dl]: a time period starting at time "est" and ending at time "dl", with a list "related-tasks" recording all tasks that fall into this range[5]*. slot.load is the sum of the process time of all related tasks.*
Slots: a list of all slots, initialized as $\emptyset$.

```
begin
    for every task t in V
        for every slot s in Slots
            exactmatch = false;
            if (s.est==t.est && s.dl == t.dl)
                exactmatch = true;
                s.load += t.process_time;
                s.relatedTasks.addElement(t);
            else if (s.est <= t.est && s.dl >= t.dl)
                s.load += t.process_time;
                s.relatedTasks.addElement(t);
            else if (s.est <= t.est && s.dl <= t.dl)
                add newslot[s.est, t.dl] to Slots;
                newslot.load = s.load + t.process_time;
                newslot.relatedTasks = union(s.relatedTasks,t);
            else if (s.est >= t.est && s.dl >= t.dl)
                add newslot[t.est, s.dl] to Slots;
                newslot.load = s.load + t.process_time;
                newslot.relatedTasks = union(s.relatedTasks,t);
```

[5] A task t falls into a range [est, dl] when this task has its earliest start time greater than or equal to "est" and its deadline is less than or equal to "dl".

```
            else if (s.est >= t.est && s.dl <= t.dl)
                exactmatch = true;
                add newslot[t.est, t.dl] to Slots;
                newslot.load = s.load + t.process_time;
                newslot.relatedTasks = union(s.relatedTasks,t);
            if (!exactmatch)
                add newslot[t.est, t.dl] to Slots;
                newslot.load = t.process_time;
                newslot.relatedTasks = union(t.label);
            if (s.load > s.dl -s.est or newslot.load > newslot.dl - newslot.est)
                print "overflow";
                return false;
        end of for slot
    end of for task
    return true;
end
```

Algorithm 2.2.6 *Find_NL_Range* Find the biggest free range for task nlt.

```
Input: A partial order schedule G(V, E) containing a task nlt;
Output: The biggest free range for task nlt;
begin
    for every task t in Pre[nlt]
        if (t.dl > nlt.est)
            t.save_dl = t.dl;
            t.dl = nlt.est;
    for every task t in Post[nlt]
        if (t.est < nlt.dl)
            t.save_est = t.est;
            t.est = nlt.dl;
    range_evaluation;
    if slot s is found to overflow
        check if there is a task t with ( t.save_dl > t.dl) or (t.save_est < t.est)
        adjust t.est or t.dl to accommodate the overflow
    if the overflow can not be solved after the adjustment
        there is no feasible schedule
    After adjustment and Range_Evaluation return true
    nlt.est = max(deadline of tasks in Pre[nlt]);
    nlt.dl = min(est of tasks in Post[nlt]);
end
```

APPENDIX C

PROOF OF THEOREM 2.2.1

Theorem 2.2.1 If there exists a feasible linear schedule, the Feasible_Schedule algorithm (Algorithm 2.2.4) can find one.

Proof Suppose the Feasible_Schedule reports failure and can not find a feasible linear schedule; the reasons could be one of the following:

1. There is no task ready for execution. It implies there is a cycle among those tasks; hence there does not exist a feasible linear schedule;

2. A task t is late for its deadline ($t.dl$).

If $t.est + t.process\_time < t.dl$, then task t is an impossible task; there is no feasible linear schedule. If $t.est + t.process\_time > t.dl$, but t is still late, it is due to the fact that t can not start at $t.est$. The are two reasons why task t can not start at $t.est$. One is that another task $t1$ with an earlier deadline has to start before task t; otherwise $t1$ would be late. In this case, either there is no feasible schedule to satisfy both $t1$ and t, or we need find why $t1$ can not start earlier, which is exactly the same thing as we are doing now for task t. The other reason that task t can not start earlier is that its pretasks have not finished by $t.est$. Suppose $t2$ is one of t's pretasks that can not finish by time $t.est$. If there exists a feasible schedule, it needs to move $t2$ earlier. To perform this move, another task $t3$ needs to be moved later. Following the above algorithm, we have:
$t3.dl <= t2.dl < t.dl$
Suppose $t3$ starts at time st in the schedule generated; the part of the schedule from task $t3$ is:
$[t3 - -t2 - -t]$ (is late)
Let us ignore other tasks in this part besides these three tasks, for it does not affect the following reasoning process.
if $(st + t3.process\_time + t2.process\_time + t.process\_time) <= t.dl$, task t would not be late following the current schedule. Since t is late, we have:
$(st + t3.process\_time + t2.process\_time + t.process\_time) > t.dl$
If we only switch task $t3$ and task $t2$, as $[t2 - -t3 - -t]$, t is still late; if we move task $t2$ after task t like $[t2 - -t - -t3]$, we have:
$(st + t2.process\_time + t.process\_time + t3.process\_time) > t.dl > t3.dl$
$t3$ is late; hence there is no way to find a feasible schedule by moving $t3$ later. So we have proved that if the Feasible_Schedule algorithm (Algorithm 2.2.4) doesn't find a feasible linear schedule, one does not exist.
END.

BIBLIOGRAPHY

[1] Andersson, M., and Sandholm, T. Leveled Commitment Contracting among Myopic Individually Rational Agents. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98)* (Paris, France, 1998), pp. 26–33.

[2] Andersson, M., and Sandholm, T. Leveled Commitment Contracts with Myopic and Strategic Agents. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence* (Madison, WI, July 1998), pp. 38–45.

[3] Andersson, M., and Sandholm, T. Time-Quality Tradeoffs in Reallocative Negotiation with Combinatorial Contract Types. pp. 3–10.

[4] Austin, J.L. *How to do Things with Words.* Harvard Univ. Press, 1962.

[5] Axelrod, Robert M. *The Evolution of Cooperation.* Basic Books, 1985.

[6] Azulay-Schwartz, R., and Kraus, S. Negotiation on data allocation in multi-agent environments. *Autonomous Agents and Multi-Agent Systems journal* (2000).

[7] Bui, H. H., Venkatesh, S., and Kieronska, D. Learning other agents' preferences in multiagent negotiation using the bayesian classifier. *International Journal of Cooperative Information Systems* (1999).

[8] Burkhard, H.D. Liveness and fairness in multi-agent systems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence.*

[9] Bussmann, S., and Muller, H.J. A negotiation framework for cooperating agents. In *Proceedings of Special Interest Group on Cooperative Knowledge Based System.*

[10] Campbell, J.A., and d'Inverno, M.P. Knowledge interchange protocols. In *Decentralized Artificial Intelligence 2.*

[11] Chang, M.K., and Woo, C.C. Sanp: A communication level protocol for negotiations. In *Decentralized Artificial Intelligence 3.*

[12] Chu-Caroll, Jennifer, and Carberry, Sandra. Response generation in collaborative negotiation. In *Proceedings of the Thirty-Third Meeting of the Association for Computational Linguistics.*

[13] Chu-Carroll, J., and Carberry, S. Conflict detection and resolution in collaborative planning. In *Intelligent Agents II.*

[14] Collins, John, Tsvetovas, Maksim, Sundareswara, Rashmi, van Tonder, Joshua, Gini, Maria, and Mobasher, Bamshad. Evaluating risk: flexibility and feasibility in multi-agent contracting. In *Proceedings of the Third International Conference on Autonomous Agents (Agents99).*

[15] Conry, S. E., Kuwabara, K., Lesser, V. R., and Meyer, R. A. Multistage negotiation for distributed constraint satisfaction. *IEEE Transactions on Systems, Man, and Cybernetics 21*, 6 (Nov. 1992).

[16] Decker, Keith S., and Lesser, Victor R. Quantitative modeling of complex environments. *International Journal of Intelligent Systems in Accounting, Finance, and Management 2*, 4 (Dec. 1993), 215–234. Special issue on "Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behavior".

[17] Deshmukh, A. V., Talavage J. J., and Barash, M. M. Complexity in manufacturing systems: Part 1 - analysis of static complexity. *IIE Transactions* (1998).

[18] Excelente-Toledo, C. B., and Jennings, N. R. Learning to select a coordination mechanism". In *Proceedings of the First Joint International Conference on Autonomous Agents and Multi-Agent Systems.*

[19] Faratin, P., Sierra, C., and Jennings, N. Negotiation Decision Functions for Autonomous Agents. *International Journal of Robotics and Autonomous Systems 24*, 3-4 (1998), 159–182.

[20] Fischer, K., Kuhn-N. Muller H.J., and Muller, J.-P. Modeling the transportation domain. *Computational Economics 8* (1995).

[21] Glass, Alyssa, and Grosz, Barbara. Socially conscious decision-making. In *Proceedings of Agents2000 Conference.*

[22] Horling, Bryan, Lesser-Victor Vincent Regis. Multi-agent system simulation framework. In *16th IMACS World Congress 2000 on Scientific Computation, Applied Mathematics and Simulation.*

[23] Horling, Bryan, Vincent-Regis Mailler Roger Shen Jiaying Becker Raphen Rawlins Kyle, and Lesser, Victor. Distributed sensor network for real time tracking. In *Proceedings of the 5th International Conference on Autonomous Agents.*

[24] Huhns, M. N., Bridgeland D. M., and Arni, N. V. A dai communication aide. In *Proceedings of the 10th International Workshop on Distributed Artificial Intelligence.*

[25] Hunsberger, Luke, and Grosz, Barbara J. A combinatorial auction for collaborative planning. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000).*

[26] John Collins, Scott Jamison, Bamshad Mobasher Maria Gin. A market architecture for multi-agent contracting. In *Proceedings of the Second International Conference on Autonomous Agents (Agents98).*

[27] Kakehi, R., and Tokoro, M. A negotiation protocol for conflict resolution in multi-agent environments. In *Proceedings of the International Conference on Intelligent Cooperative Information Systems.*

[28] Kosoresow, A.P. On the efficiency of agent-based systems. In *Proceedings of the International Conference on Autonomous Systems.*

[29] Laasri, B., Laasri H. Lander S., and Lesser, V. A generic model for intelligent negotiating agents. *International Journal on Intelligent Cooperative Information Systems* (1992).

[30] Lander, S., and Lesser, V. Understanding the role of negotiation in distributed search among heterogeneous agents. In *Proceedings of the International Joint Conference on Artificial Intelligence.*

[31] Lassri, Brigitte, Lassri, Hassan, and Lesser, Victor R. Negotiation and its role in cooperative distributed problem solving. In *Proceedings of the Tenth International Workshop on Distributed AI.*

[32] Lee, Lyndon. A model of progressive ma negotiation. In *Proceedings of the third International Conference on Multi–Agent Systems (ICMAS-98).*

[33] Lee, Lyndon C. Progressive multi-agent negotiation. In *Proceedings of the First International Conference on Multi–Agent Systems.*

[34] Lewicki, Roy J., and Litterer, Joseph A. *Negotiation.* Richard D. Irwin Inc, 1985.

[35] Matos, Noyda, Sierra, Carles, and Jennings, Nick R. Determining successful negotiation strategies: An evolutionary approach. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98)* (1998).

[36] Moehlman, T., Lesser, V., and Buteau, B. Decentralized Negotiation: An Approach to the Distributed Planning Problem. In *Group Decision and Negotiation*, K. Sycara, Ed., vol. 1. Kluwer Academic Publishers, 1992, pp. 161–192.

[37] Muller, H. J., Ed. *Foundations of Distributed Artificial Intelligence.* John Wiley and Sons, 1996, ch. 7: Negotiation Principles.

[38] N. R. Jennings, P. Faratin, T. J. Norman P. O'Brien, and Odgers, B. Autonomous agents for business process management. *International Journal of Applied Artificial Intelligence* (2000).

[39] N. R. Jennings, P. Faratin, T. J. Norman P. O'Brien B. Odgers, and Alty, J. L. Implementing a business process management system using adept: A real-world case study. *International Journal of Applied Artificial Intelligence* (2000).

[40] Oliveira, Eugenio, and Rocha, Ana Paula. Agents advanced features for negotiation in electronic commerce and virtual organizations formation process. In *Book on European perspectives on AMEC.*

[41] Pritsker, A.A.B. Gert networks: Graphical evaluation and review technique. *The Production Engineer* (1968).

[42] Pruitt, D. G. *Negotiation Behavior.* Academic Press, 1981.

[43] Rosenschein, Gilad Zlotkin Jeffrey S. A domain theory for task oriented negotiation. In *Proceedings 13th International Joint Conference on Artificial Intelligence.*

[44] Rosenschein, Gilad Zlotkin Jeffrey S. Mechanisms for automated negotiation in state oriented domains. *Journal of Artificial Intelligence Research* (1996).

[45] S. Parsons, C. Sierra, and Jennings, N. R. Agents that reason and negotiate by arguing. *Journal of Logic and Computation* (1998).

[46] Sablayrolles, P., and Schupeta, A. Conflict resolving negotiation for cooperative schedule management agents (cosma). In *DFKI Tech. Memo.*

[47] Sandholm, T., and Lesser, V. On automated contracting in multi-enterprise manufacturing. In *Proceedings of the Improving Manufacturing Performance in a Distributed Enterprise: Advanced Systems and Tools.*

[48] Sandholm, T., and Lesser, V. Advantages of a leveled commitment contracting protocol. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (1996), pp. 126–133.

[49] Sandholm, T., and Suri, S. Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. In *National Conference on Artificial Intelligence (AAAI).*

[50] Sandholm, T., and Vulkan, N. 1999. Bargaining with deadlines. In *National Conference on Artificial Intelligence (AAAI).*

[51] Sandholm, Tumas, and Lesser, Victor. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS95)* (1995).

[52] Sandholm, Tuomas W. *Multiagent Systems, A modern Approach to Distributed Artificial Intelligence.* MIT Press, 1999, ch. 5: Distributed Rational Decision Making.

[53] Sarit Kraus, Jonathan Wilkenfeld, Gilad Zlotkin. Multiagent negotiation under time constraints. *Artificial Intelligence* (1995).

[54] Sathi, Arvind. *Cooperation Through Constraint Directed Negotiation: Study of Resource Reallocation Problems.* PhD thesis, Carnegie Mellon University, 1988.

[55] Searle, J.R. *Speech Acts: An Essay in the Philosophy of Language.* Cambridge University Press, 1969.

[56] Sen, Sandip. Reciprocity: a foundational principle for promoting cooperative behavior among self-interested agents. In *Proceedings of the Second International Conference on Multiagent Systems.*

[57] Sen, Sandip, and Durfee, Edmund H. The role of commitment in cooperative negotiation. *International Journal of Intelligent and Cooperative Information Systems* (1994).

[58] Sen, Sandip, and Durfee, Edmund H. A contracting model for flexible distributed scheduling. *Annals of Operations Research* (1996).

[59] Sen, Sandip, and Durfee, Edmund H. A formal study of distributed meeting scheduling. *Group Decision and Negotiation* (1998).

[60] Smith, Reid G. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers* (1980).

[61] Sycara, Katia. Multi-agent compromise via negotiation. *Distributed Artificial Intelligence* (1989).

[62] Vincent, R.; Horling, B.; Lesser V. An agent infrastructure to build and evaluate multi-agent systems: The java agent framework and multi-agent system simulator. In *In Lecture Notes in Artificial Intelligence: Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems.*

[63] Vincent, Regis, Horling, Bryan, and Lesser, Victor. Multi-agent system simulation framework. In *16th IMACS World Congress 2000 on Scientific Computation, Applied Mathematics and Simulation* (August 2000), EPFL.

[64] von Martial, F. Coordinating plans of autonomous agents. In *Lecture Notes on Artificial Intelligence, No.610.*

[65] Wagner, Thomas, Garvey, Alan, and Lesser, Victor. Criteria-Directed Heuristic Task Scheduling. *International Journal of Approximate Reasoning, Special Issue on Scheduling 19*, 1-2 (1998), 91–118. A version also available as UMASS CS TR-97-59.

[66] Wagner, Thomas, and Lesser, Victor. Evolving real-time local agent control for large-scale mas. In *Intelligent Agents VIII (Proceedings of ATAL-01).*

[67] Wagner, Thomas, and Lesser, Victor. Relating quantified motivations for organizationally situated agents. In *Intelligent Agents VI (Proceedings of ATAL-99)*, N.R. Jennings and Y. Lespérance, Eds., Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, 2000.

[68] Weiss, Gerhard, Ed. *Multiagent Systems, A modern Approach to Distributed Artificial Intelligence.* MIT Press, 1999.

[69] Werkman, K.J. Knowledge-based model of negotiation using shareable perspectives. In *Proceedings Workshop Distributed Artificial Intelligence, 10th.*

[70] Xiaoqin Zhang, Rodion Podorozhny, Victor Lesser. Cooperative, multistep negotiation over a multi-dimensional utility function multi-agent systems negotiation. In *Proceedings of the IASTED International Conference, Artificial Intelligence and Soft Computing (ASC 2000).*

[71] Zeng, Dajun, and Sycara, Katia. Benefits of learning in negotiation. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97).*

[72] Zlotkin, G., and Rosenschein, J.S. Blocks, lies, and postal freight: The nature of deception in negotiation. In *Proceedings of the 10th International Workshop Distributed Artificial Intelligence.*

[73] Zlotkin, G., and Rosenschein, J.S. A domain theory for task oriented negotiation. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence.*

[74] Zlotkin, G., and Rosenschein, J.S. Negotiation and goal relaxation. In *Decentralized Artificial Intelligence 2.*

[75] Zlotkin, G., and Rosenschein, J.S. Negotiation with incomplete information about worth: Strict versus tolerant mechanisms. In *Proceedings of the International Conference on Intelligent and Cooperative Information Systems.*

[76] Zlotkin, Gilad, and Rosenschein, Jeffrey S. Negotiation and task sharing among autonomous agents in cooperative domains. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.*

[77] Zlotkin, Gilad, and Rosenschein, Jeffrey S. Compromise in negotiation: Exploiting worth functions over states. *Artificial Intelligence* (1996).

[78] Zlotkin, Gilad, and Rosenschein, Jeffrey S. Mechanism design for automated negotiation, and its application to task oriented domains. *Artificial Intelligence* (1996).