Meta-Level Coordination for Solving Negotiation Chains in Semi-Cooperative Multi-Agent Systems

Xiaoqin Zhang
Computer and Information Science Department
University of Massachusetts at Dartmouth
x2zhang@umassd.edu

Victor Lesser
Computer Science Department
University of Massachusetts at Amherst
lesser@cs.umass.edu

ABSTRACT

A negotiation chain is formed when multiple related negotiations are spread over multiple agents. In order to appropriately order and structure the negotiations occurring in the chain so as to optimize the expected utility, we present an extension to a single-agent concurrent negotiation framework. This work is aimed at semi-cooperative multi-agent systems, where each agent has its own goals and works to maximize its local utility; however, the performance of each individual agent is tightly related to other agent's cooperation and the system's overall performance. We introduce a pre-negotiation phase that allows agents to transfer meta-level information. Using this information, the agent can build a more accurate model of the negotiation in terms of modeling the relationship of flexibility and success probability. This more accurate model helps the agent in choosing a better negotiation solution in the global negotiation chain context. The agent can also use this information to allocate appropriate time for each negotiation, hence to find a good ordering of all related negotiations. The experimental data shows that these mechanisms improve the agents' and the system's overall performance significantly.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms, Performance, Experimentation

Keywords

Negotiation Chain, Flexibility, Multi-Linked Negotiation

1. INTRODUCTION

Sophisticated negotiation for task and resource allocation is crucial for the next generation of multi-agent systems (MAS) applications. Groups of agents need to efficiently negotiate over multiple related issues concurrently in a complex, distributed setting where there are deadlines by which the negotiations must be completed. This is an important research area where there has been very little work done.

This work is aimed at **semi-cooperative multi-agent systems**, where each agent has its own goals and works to maximize its local utility; however, the performance of each individual agent is tightly related to other agent's cooperation and the system's overall performance. There is no single global goal in such systems, either because each agent represents a different organization/user, or because it is difficult/impossible to design one single global goal. This issue arises due to multiple concurrent tasks, resource constrains and uncertainties, and thus no agent has sufficient knowledge or computational resources to determine what is best for the whole system [11]. An example of such a system would be a virtual organization [12] (i.e. a supply chain) dynamically formed in an electronic marketplace such as the one developed by the CONOISE project [5]. To accomplish tasks continuously arriving in the virtual organization, cooperation and sub-task relocation are needed and preferred. There is no single global goal since each agent may be involved in multiple virtual organizations. Meanwhile, the performance of each individual agent is tightly related to other agents' cooperation and the virtual organization's overall performance. The negotiation in such systems is **not a zero-sum game**, a deal that increases both agents' utilities can be found through efficient negotiation. Additionally, there are multiple encounters among agents since new tasks are arriving all the time. In such negotiations, price may or may not be important, since it can be fixed resulting from a long-term contract. Other factors like quality and delivery time are important too. Reputation mechanisms in the system makes cheating not attractive from a long term viewpoint due to multiple encounters among agents. In such systems, agents are **self-interested** because they primarily focus on their own goals; but they are also **semi-cooperative**, meaning they are willing to be truthful and collaborate with other agents to find solutions that are beneficial to all participants, including itself; though it won't voluntarily scarify its own utility in exchange of others' benefits.

Another major difference between this work and other work on negotiation is that negotiation, here, is not viewed as a stand-alone process. Rather it is 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 work on negotiation is concerned more about the meta-level decision-making process in negotiation rather than the basic protocols or languages. The goal of this research is to develop a set of **macro-strategies** that allow the agents to effectively manage multiple related negotiations, including, but not limited to the following issues: how much time should be spent on each negotiation, how much *flexibility* (see formal definition in Formula 3) should be allocated for each negotiation, and in what order should

AAMAS'07 May 14–18 2007, Honolulu, Hawai'i, USA.

978-81-904262-7-5 (RPS)

the negotiations be performed. These macro-strategies are different from those **micro-strategies** that direct the individual negotiation thread, such as whether the agent should concede and how much the agent should concede, etc[3].

In this paper we extend a multi-linked negotiation model [10] from a single-agent perspective to a multi-agent perspective, so that a group of agents involved in chains of interrelated negotiations can find nearly-optimal macro negotiation strategies for pursuing their negotiations. The remainder of this paper is structured in the following manner. Section 2 describes the basic negotiation process and briefly reviews a single agent's model of multi-linked negotiation. Section 3 introduces a complex supply-chain scenario. Section 4 details how to solve those problems arising in the negotiation chain. Section 5 reports on the experimental work. Section 6 discusses related work and Section 7 presents conclusions and areas of future work.

2. BACKGROUND ON MULTI-LINKED NEGOTIATION

In this work, the negotiation process between any pair of agents is based on an extended version of the contract net [6]: the initiator agent announces the proposal including multiple features; the responding agent evaluates it and responds with either a yes/no answer or a counter proposal with some features modified. This process can go back and forth until an agreement is reached or the agents decide to stop. If an agreement is reached and one agent cannot fulfill the commitment, it needs to pay the other party a decommitment penalty as specified in the commitment. A negotiation starts with a proposal, which announces that a task (t) needs to be performed includes the following attributes:

1. *earliest start time* (est): the earliest start time of task t; task t cannot be started before time est.
2. *deadline* (dl): the latest finish time of the task; the task needs to be finished before the deadline dl.
3. *minimum quality requirement* ($minq$): the task needs to be finished with a quality achievement no less than $minq$.
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 earlier than *dl*, it will get the *extra early finish reward* proportional to this rate.
6. *decommitment penalty rate* (p): if the contractor agent cannot perform the task as it promised in the contract or if the contractee agent needs to cancel the contract after it has been confirmed, it also needs to pay a *decommitment penalty* ($p*r$) to the other agent.

The above attributes are also called attribute-**in**-negotiation which are the features of the subject (issue) to be negotiated, and they are domain-dependent. Another type of attribute [1] is the attribute-**of**-negotiation, which describes the negotiation process itself and is domain-independent, such as:

1. negotiation duration ($\delta(v)$): the maximum time allowed for negotiation v to complete, either reaching an agreed upon proposal (success) or no agreement (failure).
2. negotiation start time ($\alpha(v)$): the start time of negotiation v. $\alpha(v)$ is an attribute that needs to be decided by the agent.
3. negotiation deadline ($\epsilon(v)$): negotiation v needs to be finished before this deadline $\epsilon(v)$. The negotiation is no longer valid after time $\epsilon(v)$, which is the same as a failure outcome of this negotiation.
4. success probability ($p_s(v)$): the probability that v is successful. It depends on a set of attributes, including both attributes-in-negotiation (i.e. reward, flexibility, etc.) and attributes-of-negotiation (i.e. negotiation start time, negotiation deadline, etc.).

An agent involved in multiple related negotiation processes needs to reason on how to manage these negotiations in terms of ordering them and choosing the appropriate values for features. This is the **multi-linked negotiation problem** [10] :

DEFINITION 2.1. *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 negotiations, and $E = \{(u, v)\}$ is a set of binary relations on V. $(u, v) \in E$ denotes that negotiation u and negotiation v are directly-linked. The relationships among the negotiations are described by a forest, a set of rooted trees $\{T_i\}$. There is a relation operator associated with every non-leaf negotiation v (denoted as $\rho(v)$), which describes the relationship between negotiation v and its children. This relation operator has two possible values: AND and OR. The AND relationship associated with a negotiation v means the successful accomplishment of the commitment on v requires all its children nodes have successful accomplishments. The OR relationship associated with a negotiation v means the successful accomplishment of the commitment on v requires at least one child node have successful accomplishment, where the multiple children nodes represent alternatives to accomplish the same goal.*

Multi-linked negotiation problem is a local optimization problem. To solve a multi-linked negotiation problem is to find a negotiation solution (ϕ, φ) with optimized expected utility $\mathcal{EU}(\phi, \varphi)$, which is defined as:

$$\mathcal{EU}(\phi, \varphi) = \sum_{i=1}^{2^n} P(\chi_i, \varphi) * (R(\chi_i, \varphi) - C(\chi_i, \phi, \varphi)) \qquad (1)$$

A **negotiation ordering** ϕ defines a partial order of all negotiation issues. A **feature assignment** φ is a mapping function that assigns a value to each attribute that needs to be decided in the negotiation. A negotiation outcome χ for a set of negotiations $\{v_j\}, (j = 1, ..., n)$ specifies the result for each negotiation, either success or failure. There are a total of 2^n different outcomes for n negotiations: $\{chi_i\}, (i = 1, ..., 2^n)$. $P(\chi_i, \varphi)$ denotes the probability of the outcome χ_i given the feature assignment φ, which is calculated based on the success probability of each negotiation. $R(\chi_i, \varphi)$ denotes the agent's utility increase given the outcome χ_i and the feature assignment φ, and $C(\chi_i, \phi, \varphi)$ is the sum of the decommitment penalties of those negotiations, which are successful, but need to be abandoned because the failure of other directly related negotiations; these directly related negotiations are performed concurrently with this negotiation or after this negotiation according to the negotiation ordering ϕ.

[1] These attributes are similar to those used in project management; however, **the multi-linked negotiation problem cannot be reduced to a project management problem or a scheduling problem**. The multi-linked negotiation problem has two dimensions: the negotiations, and the subjects of negotiations. The negotiations are interrelated and the subjects are interrelated; the attributes of negotiations and the attributes of the subjects are interrelated as well. This two-dimensional complexity of interrelationships distinguishes it from the classic project management problem or scheduling problem, where all tasks to be scheduled are local tasks and no negotiation is needed.

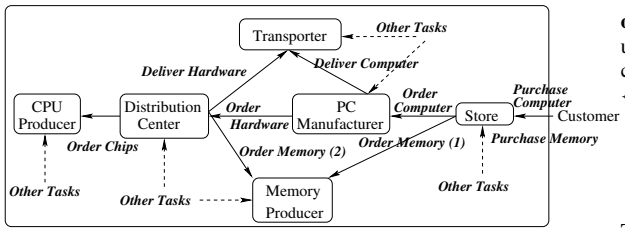

Figure 1: A Complex Negotiation Chain Scenario

A heuristic search algorithm [10] has been developed to solve the single agent's multi-linked negotiation problem that produces nearly-optimal solutions. This algorithm is used as the core of the decision-making for each individual agent in the negotiation chain scenario. In the rest of the paper, we present our work on how to improve the local solution of a single agent in the global negotiation chain context.

3. NEGOTIATION CHAIN PROBLEM

Negotiation chain problem occurs in a multi-agent system, where each agent represents an individual, a company, or an organization, and there is no absolute authority in the system. Each agent has its own utility function for defining the implications of achieving its goals. The agent is designed to optimize its expected utility given its limited information, computational and communication resources. Dynamic tasks arrive to individual agents, most tasks requiring the coordination of multiple agents. Each agent has the scheduling and planning ability to manage its local activities, some of these activities are related to other agents' activities. Negotiation is used to coordinate the scheduling of these mutual related activities. The negotiation is tightly connected with the agent's local scheduling/planning processes and is also related to other negotiations. An agent may be involved in multiple related negotiations with multiple other agents, and each of the other agents may be involved in related negotiations with others too.

Figure 1 describes a complex negotiation chain scenario. The Store, the PC manufacturer, the Memory Producer and the Distribution Center are all involved in multi-linked negotiation problems. Figure 2 shows a distributed model of part of the negotiation chain described in Figure 1. Each agent has a local optimization problem - the multi-linked negotiation problem (represented as an and-or tree), which can be solved using the model and procedures described in Section 2. However, the local optimal solution may not be optimal in the global context given the local model is neither complete or accurate. The dash line in Figure 2 represents the connection of these local optimization problem though the common negotiation subject.

Negotiation chain problem O is a group of tightly-coupled local optimization problems:

$O = \{O_1, O_2, ...O_n\}$, O_i denotes the local optimization problem (multi-linked negotiation problem) of agent A_i

Agent A_i's **local optimal** solution S_i^{lo} maximizes the expected local utility based on an incomplete information and assumptions about other agents' local strategies - we defined such incomplete information and imperfect assumptions of agent i as I_i):

$$U_i^{exp}(S_i^{lo}, I_i) \geq U_i^{exp}(S_i^x, I_i) \text{ for all } x \neq lo.$$

However, the combination of these local optimal solutions $\{S_i^{lo}\}$: $< S_1^{lo}, S_2^{lo}, ...S_n^{lo} >$ can be sub-optimal to a set of **better local optimal** solutions $\{S_i^{blo}\} : < S_1^{blo}, S_2^{blo}, ...S_n^{blo} >$ if the global utility can be improved without any agent's local utility being decreased by using $\{S_i^{blo}\}$. In other words, $\{S_i^{lo}\}$ is *dominated by* $\{S_i^{blo}\}$ ($\{S_i^{lo}\} \prec \{S_i^{blo}\}$) **iff**:

$$U_i(< S_1^{lo}, S_2^{lo}, ...S_n^{lo} >) \leq U_i(< S_1^{blo}, S_2^{blo}, ...S_n^{blo} >) \text{ for } i = 1, ...n \textbf{ and } \sum_{i=1}^{n} U_i(< S_1^{lo}, S_2^{lo}, ...S_n^{lo} >) < \sum_{i=1}^{n} U_i(< S_1^{blo}, S_2^{blo}, ...S_n^{blo} >)$$

There are multiple sets of better local optimal solutions: $\{S_i^{blo_1}\}$, $\{S_i^{blo_2}\}$, ... $\{S_i^{blo_m}\}$. Some of them may be dominated by others. A set of better local optimal solutions $\{S_i^{blo_g}\}$ that is not dominated by any others is called **best local optimal**. If a set of best local optimal solutions $\{S_i^{blo_g}\}$ dominates all others, $\{S_i^{blo_g}\}$ is called **globally local optimal**. However, sometimes the globally local optimal set does not exist, instead, there exist multiple sets of best local optimal solutions. Even if the globally local optimal solution does exist in theory, finding it may not be realistic given the agents are making decision concurrently, to construct the perfect local information and assumptions about other agents (I_i) in this dynamic environment is a very difficult and sometimes even impossible task.

The goal of this work is to improve each agent's local model about other agents (I_i) through meta-level coordination. As I_i become more accurate, the agent's local optimal solution to its local multi-linked negotiation problem become a better local optimal solution in the context of the global negotiation chain problem. We are not arguing that this statement is a universal valid statement that holds in all situations, but our experimental work shows that the sum of the agents' utilities in the system has been improved by 95% on average when meta-level coordination is used to improve each agent's local model I_i. In this work, we focus on improving the agent's local model through two directions. One direction is to build a better function to describe the relationship between the success probability of the negotiation and the flexibility allocated to the negotiation. The other direction is to find how to allocate time more efficiently for each negotiation in the negotiation chain context.

4. NEW MECHANISM - META-LEVEL COORDINATION

In order for an agent to get a better local model about other agents in the negotiation chain context, we introduce a *pre-negotiation* phase into the local negotiation process. During the pre-negotiation phase, agents communicate with other agents who have tasks contracting relationships with them, they transfer meta-level information before they decide on how and when to do the negotiations. Each agent tells other agents what types of tasks it will ask them to perform, and the probability distributions of some parameters of those tasks, i.e. the earliest start times and the deadlines, etc. When these probability distributions are not available directly, agents can learn such information from their past experience. In our experiment described later, such distributed information is learned rather than being directly told by other agents. Specifically, each agent provides the following information to other related agents:

- Whether additional negotiation is needed in order to make a decision on the contracting task; if so, how many more negotiations are needed. **negCount** represents the total number of additional negotiations needed for a task, including additional negotiations needed for its subtasks that happen among other agents. In a negotiation chain situation, this information is being propagated and updated through the chain until

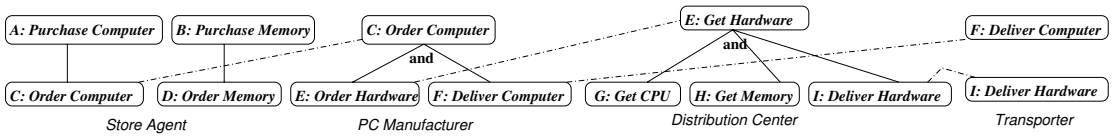

Figure 2: Distributed Model of Negotiation Chains

every agent has accurate information. Let $subNeg(T)$ be a set of subtasks of task T that require additional negotiations, then we have:

$$negCount(T) = |subNeg(T)| + \sum_{t \in subNeg(T)} (negCount(t)) \quad (2)$$

For example, in the scenario described in Figure 1, for the distribution center, task *Order Hardware* consists of three subtasks that need additional negotiations with other agents: *Order Chips*, *Order Memory* and *Deliver Hardware*. However, no further negotiations are needed for other agents to make decision on these subtasks, hence the $negCount$ for these subtasks are 0. The following information is sent to the PC manufacturer by the distribution center:
$negCount(Order\_Hardware) = 3$
For the PC manufacturer task *Order Computer* contains two subtasks that requires additional negotiations: *Deliver Computer* and *Order Hardware*. When the PC manufacturer receives the message from the Distribution Center, it updates its local information:
$negCount(Order\_Computer) = 2+$
$negCount(Deliver\_Computer)(0)+$
$negCount(Order\_Hardware)(3) = 5$
and sends the updated information to the Store Agent.

- Whether there are other tasks competing with this task and what is the likelihood of conflict. Conflict means that given all constrains, the agent cannot accomplish all tasks on time, it needs to reject some tasks. The likelihood of conflict Pc_{ij} between a task of type i and another task of type j is calculated based on the statistical model of each task's parameters, including earliest start time (est), deadline (dl), task duration (dur) and slack time (sl), using a formula [7]: $Pc_{ij} = P(dl_i - est_j \leq dur_i + dur_j \wedge dl_j - est_i \leq dur_i + dur_j)$

  When there are more than two types of tasks, the likelihood of no conflict between task i and the rest of the tasks, is calculated as: $P_{noConflict}(i) = \prod_{j=1,j\neq i}^{n}(1 - Pc_{ij})$

For example, the Memory Producer tells the Distribution Center about the task *Order Memory*. Its local decision does not involve additional negotiation with other agents ($negCount = 0$), however, there is another task from the Store Agent that competes with this task, thus the likelihood of no conflict is 0.5 ($P_{noConflict} = 0.5$). On the other hand, the CPU Producer tells the Distribution Center about the task *Order Chips*: its local decision does not involve additional negotiation with other agents, and there are no other tasks competing with this task ($P_{noConflict} = 1.0$) given the current environment setting. Based on the above information, the Distribution Center knows that task *Order Memory* needs more flexibility than task *Order Chips* in order to be successful in negotiation. Meanwhile, the Distribution Center would tell the PC Manufacturer that task *Order Hardware* involves further negotiation with other agents ($negCount = 3$), and that its local decision depends on other agents' decisions. This piece of information helps the PC Manufacturer allocate appropriate flexibility for task *Order Hardware* in negotiation. In this work, we introduce a short period

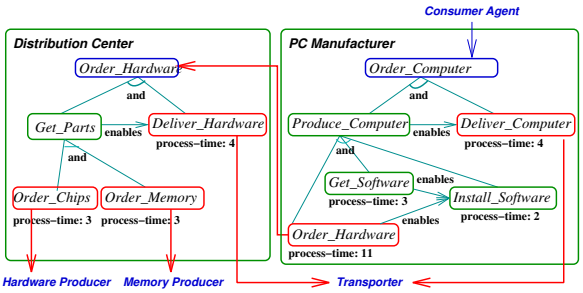

Figure 3: Task Structures of PC Manufacturer and Distribution Center

for agents to learn the characteristics of those incoming tasks, including est, dl, dur and sl, which are used to calculate Pc_{ij} and $P_{noConflict}$ for the meta-level coordination. During system performance, agents are continually monitoring these characteristics. An updated message will be send to related agents when there is significant change of the meta-level information.

Next we will describe how the agent uses the meta-level information transferred during the pre-negotiation phase. This information will be used to improve the agent's local model, more specifically, they are used in the agent's local decision-making process by affecting the values of some features. Especially, we will be concerned with two features that have strong implications for the agent's macro strategy for the multi-linked negotiations, and hence also affect the performance of a negotiation chain significantly. The first is the *amount of flexibility* specified in the negotiation parameter. The second feature we will explore is the time allocated for the negotiation process to complete. The time allocated for each negotiation affects the possible ordering of those negotiations, and it also affects the negotiation outcome. Details are discussed in the following sections.

4.1 Flexibility and Success Probability

Agents not only need to deal with complex negotiation problems, they also need to handle their own local scheduling and planning process that are interleaved with the negotiation process. Figure 3 shows the local task structures of the PC Manufacturer and the Distribution Center. Some of these tasks can be performed locally by the PC manufacturer, such as *Get Software* and *Install Software*, while other tasks (non-local tasks) such as *Order Hardware* and *Deliver Computer* need to be performed by other agents.The PC Manufacturer needs to negotiate with the Distribution Center and the Transporter about whether they can perform these tasks, and if so, when and how they will perform them.

When the PC Manufacturer negotiates with other agents about the non-local task, it needs to have the other agents' arrangement fit into its local schedule. Since the PC Manufacturer is dealing with multiple non-local tasks simultaneously, it also needs to ensure the commitments on these non-local tasks are consistent with each other. For example, the deadline of task *Order Hardware* cannot be later than the start time of task *Deliver Computer*. Figure 4

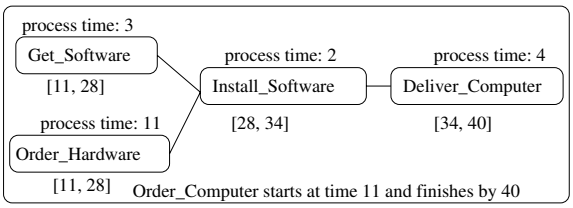

Figure 4: A Sample Local Schedule of the PC Manufacturer

shows a sample local schedule of the PC Manufacturer. According to this schedule, as long as task *Order Hardware* is performed during time [11, 28] and task *Deliver Computer* is performed during time [34, 40], there exists a feasible schedule for all tasks and task *Order Computer* can be finished by time 40, which is the deadline promised to the Customer. These time ranges allocated for task *Order Hardware* and task *Deliver Computer* are called *consistent ranges*; the negotiations on these tasks can be performed independently within these ranges without worrying about conflict. Notice that each task should be allocated with a time range that is large enough to accommodate the estimated task process time. The larger the range is, the more likely the negotiation will succeed, because it is easier for the other agent to find a local schedule for this task. Then the question is, how big should this time range be? We defined a quantitative measure called *flexibility*:

Given a task t, suppose the allocated time range for t is $[est, dl]$, est is the earliest start time and dl stands for the deadline,

$$flexibility(t) = \frac{dl - est - process\_time(t)}{process\_time(t)} \quad (3)$$

Flexibility is an important attribute because it directly affects the possible outcome of the negotiation. The success probability of a negotiation can be described as a function of the flexibility. In this work, we adopt the following formula for the success probability function based on the flexibility of the negotiation issue:

$$p_s(v) = p_{bs}(v) * (2/\pi) * (\arctan(f(v) + c))) \quad (4)$$

This function describes a phenomenon where initially the likelihood of a successful negotiation increases significantly as the flexibility grows, and then levels off afterward, which mirrors our experience from previous experiments. p_{bs} is the *basic success probability* of this negotiation v when the flexibility $f(v)$ is very large. c is a parameter used to adjust the relationship. Different function patterns can result from different parameter values, as shown in Figure 5. This function describes the agent's assumption about how the other agent involved in this negotiation would response to this particular negotiation request, when it has flexibility $f(v)$. This function is part of the agent's local model about other agents. To improve the accuracy of this function and make it closer to the reality, the agent adjusts these two values according to the meta-level information transferred during pre-negotiation phase. The values of c depends on whether there is further negotiation involved and whether there are other tasks competing with this task for common resources. If so, more flexibility is needed for this issue and hence c should be assigned a smaller value. In our implementation, the following procedure is used to calculate c based on the meta-level information $negCount$ and $P_{noConflict}$:

if($P_{noConflict} > 0.99$) // no other competing task
  $c = C_{large} - negCount$
else // competing task exists
  $c = C_{small}$

This procedure works as follows: when there is no other competing

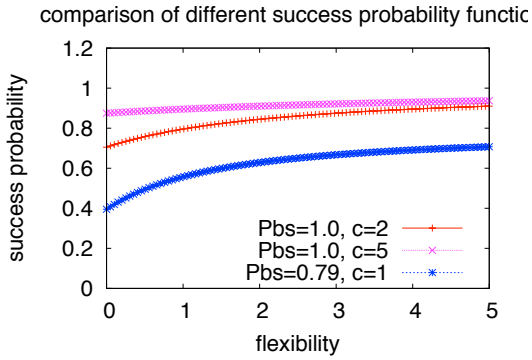

Figure 5: Different Success Probability Functions

task, c depends on the number of additional negotiations needed. The more additional negotiations that are needed, the smaller value c has, hence more flexibility will be assigned to this issue to ensure the negotiation success. If no more negotiation is needed, c is assigned to a large number C_{large}, meaning that less flexibility is needed for this issue. When there are other competing tasks, c is assigned to a small number C_{small}, meaning that more flexibility is needed for this issue. In our experimental work, we have C_{large} as 5 and C_{small} as 1. These values are selected according to our experience; however, a more practical approach is to have agents learn and dynamically adjust these values. This is also part of our future work.

p_{bs} is calculated based on $P_{noConflict}$, $f(v)$ (the flexibility of v in previous negotiation), and c, using the reverse format of equation 4.

$$p_{bs}(v) = min(1.0, P_{noConflict}(v) * (\pi/2)/(\arctan(f(v) + c))) \quad (5)$$

For example, based on the scenario described above, the agents have the following values for c and p_{bs} based on the meta-level information transferred:

- PC Manufacturer, *Order Hardware*: $p_{bs} = 1.0, c = 2$;
- Distribution Center, *Order Chips*: $p_{bs} = 1.0, c = 5$;
- Store Agent, *Order Memory*: $p_{bs} = 0.79, c = 1$;

Figure 5 shows the different patterns of the success probability function given different parameter values. Based on such patterns, the Store Agent would allocate more flexibility to the task *Order Memory* to increase the likelihood of success in negotiation. In the agent's further negotiation process, formula 4 with different parameter values is used in reasoning on how much flexibility should be allocated to a certain issue.

The pre-negotiation communication occurs before negotiation, but not before every negotiation session. Agents only need to communicate when the environment changes, for example, new types of tasks are generated, the characteristics of tasks changes, the negotiation partner changes, etc. If no major change happens, the agent can just use the current knowledge from previous communications. The communication and computation overhead of this pre-negotiation mechanism is very small, given the simple information collection procedure and the short message to be transferred. We will discuss the effect of this mechanism in Section 5.

4.2 Negotiation Duration and Deadline

In the agent's local model, there are two attributes that describe how soon the agent expects the other agent would reply to the negotiation v: negotiation duration $\delta(v)$ and negotiation deadline $\epsilon(v)$

Table 1: Examples of negotiations ($\delta(v)$: negotiation duration, s.p.: success probability)

index	task-name	$\delta(v)$	reward	s.p.	penalty
1	Order Hardware	4	6	0.99	3
2	Order Chips	4	1	0.99	0.5
3	Order Memory	4	1	0.80	0.5
4	Deliver Hardware	4	1	0.70	0.5

. These two important attributes that affect the negotiation solution. Part of the negotiation solution is a negotiation ordering ϕ which specifies in what order the multiple negotiations should be performed. In order to control the negotiation process, every negotiation should be finished before its negotiation deadline, and the negotiation duration is the time allocated for this negotiation. If a negotiation cannot be finished during the allocated time, the agent has to stop this negotiation and consider it as a failure. The decision about the negotiation order depends on the success probability, reward, and decommitment penalty of each negotiation. A good negotiation order should reduce the risk of decommitment and hence reduce the decommitment penalty. A search algorithm has been developed to find such negotiation order described in [10].

For example, Table 1 shows some of the negotiations for the Distribution Center and their related attributes. Given enough time (negotiation deadline is greater than 16), the best negotiation order is: $4 \rightarrow 3 \rightarrow 2 \rightarrow 1$. The most uncertain negotiation (4: Deliver Hardware) is performed first. The negotiation with highest penalty (1: Order hardware) is performed after all related negotiations (2, 3, and 4) have been completed so as to reduce the risk of decommitment. If the negotiation deadline is less than 12 and greater than 8, the following negotiation order is preferred: $(4, 3, 2) \rightarrow 1$, which means negotiation 4, 3, 2 can be performed in parallel, and 1 needs to be performed after them. If the negotiation deadline is less than 8, then all negotiations have to be performed in parallel, because there is no time for sequencing negotiations.

In the original model for single agent [10], the negotiation deadline $\epsilon(v)$ is assumed to be given by the agent who initiates the contract. The negotiation duration $\delta(v)$ is an estimation of how long the negotiation takes based on experience. However, the situation is not that simple in a negotiation chain problem. Considering the following scenario. When the customer posts a contract for task *Purchase Computer*, it could require the Store Agent to reply by time 20. Time 20 can be considered as the negotiation deadline for *Purchase Computer*. When the Store Agent negotiates with the PC Manufacturer about *Order Computer*, what negotiation deadline should it specify? How long the negotiation on *Order Computer* takes depends on how the PC Manufacturer handles its local multiple negotiations: whether it replies to the Store Agent first or waits until all other related negotiations have been settled. However, the ordering of negotiations depends on the negotiation deadline on *Order Computer*, which should be provided by the Store Agent. The negotiation deadline of *Order Computer* for the PC Manufacturer is actually decided based on the negotiation duration of *Order Computer* for the Store Agent. How much time the Store Agent would like to spend on the negotiation *Order Computer* is its duration, and also determines the negotiation deadline for the PC Manufacturer.

Now the question arises: how should an agent decide how much time it should spend on each negotiation, which actually affects the other agents' negotiation decisions. The original model does not handle this question since it assumes the negotiation duration $\delta(v)$ is known. Here we propose three different approaches to handle this issue.

1. *same-deadline policy*. Use the same negotiation deadline for all related negotiations, which means allocate all available time to all negotiations:
   $\delta(v) = total\_available\_time$
   For example if the negotiation deadline for *Purchase Computer* is 20, the Store Agent will tell the PC Manufacturer to reply by 20 for *Order Computer* (ignoring the communication delay). This strategy allows every negotiation to have the largest possible duration, however it also eliminates the possibility of performing negotiations in sequence - all negotiations need to be performed in parallel because the total available time is the same as the duration of each negotiation.
2. *meta-info-deadline policy*. Allocate time for each negotiation according to the meta-level information transferred in the pre-negotiation phase. A more complicated negotiation, which involves further negotiations, should be allocated additional time. For example, the PC Manufacturer allocates a duration of 12 for the negotiation *Order Hardware*, and a duration of 4 for *Deliver Computer*. The reason is that the negotiation with the Distribution Center about *Order Hardware* is more complicated because it involves further negotiations between the Distribution Center and other agents. In our implementation, we use the following procedure to decide the negotiation duration $\delta(v)$:

   if($negCount(v) >= 3$) // more additional negotiation needed
   $\delta(v) = (negCount(v)-1)*basic\_neg\_cycle$
   else if($negCount(v) > 0$) // one or two additional negotiations needed
   $\delta(v) = 2 * basic\_neg\_cycle$
   else //no additional negotiation
   $\delta(v) = basic\_neg\_cycle + 1$

   $basic\_neg\_cycle$ represents the minimum time needed for a negotiation cycle (proposal-think-reply), which is 3 in our system setting including communication delay. One additional time unit is allocated for the simplest negotiation because it allows the agent to perform a more complicated reasoning process in thinking. Again, the structure of this procedure is selected according to experience, and it can be learned and adjusted by agents dynamically.
3. *evenly-divided-deadline policy*. Evenly divide the available time among the n related negotiations:
   $\delta(v) = total\_available\_time/n$
   For example, if the current time is 0, and the negotiation deadline for *Order Computer* is 21, given two other related negotiations, *Order Hardware* and *Deliver Computer*, each negotiation is allocated with a duration of 7.

Intuitively we feel the strategy 1 may not be a good one, because performing all negotiations in parallel would increase the risk of decommitment and hence also decommitment penalties. However, it is not very clear how strategy 2 and 3 perform, and we will discuss some experimental results in Section 5.

5. EXPERIMENTS

To verify and evaluate the mechanisms presented for the negotiation chain problem, we implemented the scenario described in Figure 1 . New tasks were randomly generated with decommitment penalty rate $p \in [0, 1]$, early finish reward rate $e \in [0, 0.3]$, and deadline $dl \in [10, 60]$ (this range allows different flexibilities available for those sub-contracted tasks), and arrived at the store agent periodically. We performed two sets of experiments to study

Table 2: Parameter Values Without/With Meta-level Information

	fixed-flex	meta-info-flex	
negotiation	p_{bs}	p_{bs}	c
Order Computer	0.95	1.0	0
Order Memory (1)	0.95	0.79	1
Order Hardware	0.95	1.0	2
Deliver Computer	0.95	1.0	1
Deliver Hardware	0.95	1.0	5
Order Chips	0.95	1.0	1
Order Memory (2)	0.95	0.76	1

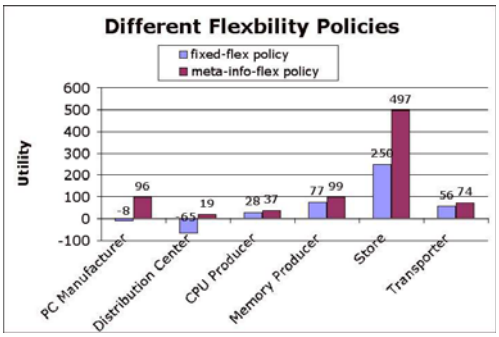

Figure 6: Different Flexibility Policies

how the success probability functions and negotiation deadlines affect the negotiation outcome, the agents' utilities and the system's overall utility. In this experiment, agents need to make decision on negotiation ordering and feature assignment for multiple attributes including: *earliest start time*, *deadline*, *promised finish time*, and those attributes-**of**-negotiation. To focus on the study of flexibility, in this experiment, the *regular reward*s for each type of tasks are fixed and not under negotiation. Here we only describe how agents handle the negotiation duration and negotiation deadlines because they are the attributes affected by the pre-negotiation phase. All other attributes involved in negotiation are handled according to how they affect the feasibility of local schedule (time-related attributes) and how they affect the negotiation success probability (time and cost related attributes) and how they affect the expect utility. A search algorithm [10] and a set of partial order scheduling algorithms are used to handle these attributes.

We tried two different flexibility policies.

1. *fixed-flexibility policy*: the agent uses a fixed value as the success probability ($p_s(v) = p_{bs}(v)$), according to its local knowledge and estimation.
2. *meta-info-flexibility policy*: the agent uses the function $p_s(v) = p_{bs}(v) * (2/\pi) * (\arctan(f(v) + c))$ to model the success probability. It also adjusts those parameters ($p_{bs}(v)$ and c) according to the meta-level information obtained in pre-negotiation phase as described in Section 4. Table 2 shows the values of those parameters for some negotiations.

Figure 6 shows the results of this experiment. This set of experiments includes 10 system runs, and each run is for 1000 simulating time units. In the first 200 time units, agents are learning about the task characteristics, which will be used to calculate the conflict probabilities Pc_{ij}. At time 200, agents perform meta-level information communication, and in the next 800 time units, agents use the meta-level information in their local reasoning process. The data was collected over the 800 time units after the pre-negotiation

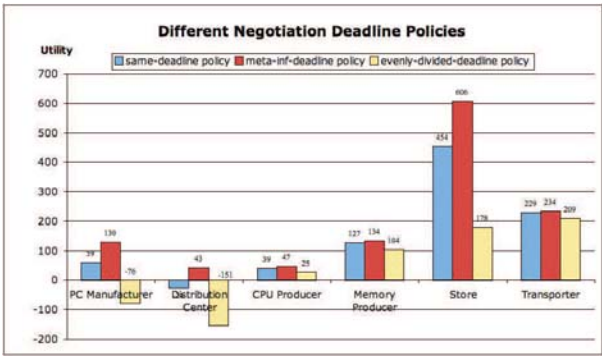

Figure 7: Different Negotiation Deadline Policies

phase [2]. One *Purchase Computer* task is generated every 20 time units, and two *Purchase Memory* tasks are generated every 20 time units. The deadline for task *Purchase Computer* is randomly generated in the range of [30, 60], the deadline for task *Purchase Memory* is in the range of [10, 30]. The decommitment penalty rate is randomly generated in the range of [0, 1]. This setting creates **multiple concurrent negotiation chain** situations; there is one long chain:

Customer - Store - PC Manufacturer - Distribution Center - Producers - Transporter

and two short chains:

Customer - Store - Memory Producer

This demonstrates that this mechanism is capable of handling multiple concurrent negotiation chains.

All agents perform better in this example (gain more utility) when they are using the meta-level information to adjust their local control through the parameters in the success probability function (meta-info-flex policy). Especially for those agents in the middle of the negotiation chain, such as the PC Manufacturer and the Distribution Center, the flexibility policy makes a significant difference. When the agent has a better understanding of the global negotiation scenario, it is able to allocate more flexibility for those tasks that involve complicated negotiations and resource contentions. Therefore, the success probability increases and fewer tasks are rejected or canceled (90% of the tasks have been successfully negotiated over when using meta-level information, compared to 39% when no pre-negotiation is used), resulting in both the agent and the system achieving better performance.

In the second set of experiments studies, we compare three negotiation deadline policies described in Section 4.2 when using the meta-info flexibility policy described above. The initial result shows that the same-deadline policy and the meta-info-deadline policy perform almost the same when the amount of system workload level is moderate, tasks can be accommodated given sufficient flexibility. In this situation, with either of the policies, most negotiations are successful, and there are few decommitment occurrences, so the ordering of negotiations does not make too much difference. Therefore, in this second set of experiments, we increase the number of new tasks generated to raise the average workload in the system. One *Purchase Computer* task is generated every 15 time units, three *Purchase Memory* tasks are generated every

[2]We only measure the utility collected after the learning phase because that the learning phase is relatively short comparing to the evaluation phase, also during the learning phase, no meta-level information is used, so some of the policies are invalid.

15 time units, and one task *Deliver Gift* (directly from the customer to the Transporter) is generated every 10 time units. This setup generates a higher level of system workload, which results in some tasks not being completed no matter what negotiation ordering is used. In this situation, we found the meta-info-deadline policy performs much better than same-deadline policy (See Figure 7). When an agent uses the same-deadline policy, all negotiations have to be performed in parallel. In the case that one negotiation fails, all related tasks have to be canceled, and the agent needs to pay multiple decommitment penalties. When the agent uses the meta-info-deadline policy, complicated negotiations are allocated more time and, correspondingly, simpler negotiations are allocated less time. This also has the effect of allowing some negotiations to be performed in sequence. The consequence of sequencing negotiation is that, if there is failure, an agent can simply cancel the other related negotiations that have not been started. In this way, the agent does not have to pay decommitment penalty for those canceled negotiations because no commitment has been established yet. The evenly-divided-deadline policy performs much worse than the meta-info-deadline policy. In the evenly-divided-deadline policy, the agent allocates negotiation time evenly among the related negotiations, hence the complicated negotiation does not get enough time to complete.

The above experiment results show that the meta-level information transferred among agents during the pre-negotiation phase is critical in building a more accurate model of the negotiation problem. The reasoning process based on this more accurate model produces an efficient negotiation solution, which improves the agent's and the system's overall utility significantly. This conclusion holds for those environments where the system is facing moderate heavy load and tasks have relatively tight time deadline (our experiment setup is to produce such environment); the efficient negotiation is especially important in such environments.

6. RELATED WORK

Fatima, Wooldridge and Jennings [1] studied the multiple issues in negotiation in terms of the agenda and negotiation procedure. However, this work is limited since it only involves a single agent's perspective without any understanding that the agent may be part of a negotiation chain. Mailler and Lesser [4] have presented an approach to a distributed resource allocation problem where the negotiation chain scenario occurs. It models the negotiation problem as a distributed constraint optimization problem (DCOP) and a cooperative mediation mechanism is used to centralize relevant portions of the DCOP. In our work, the negotiation involves more complicated issues such as reward, penalty and utility; also, we adopt a distribution approach where no centralized control is needed. A mediator-based partial centralized approach has been applied to the coordination and scheduling of complex task network [8], which is different from our work since the system is a complete cooperative system and individual utility of single agent is not concerned at all. A combinatorial auction [2, 9] could be another approach to solving the negotiation chain problem. However, in a combinatorial auction, the agent does not reason about the ordering of negotiations. This would lead to a problem similar to those we discussed when the same-deadline policy is used.

7. CONCLUSION AND FUTURE WORK

In this paper, we have solved negotiation chain problems by extending our multi-linked negotiation model from the perspective of a single agent to multiple agents. Instead of solving the negotiation chain problem in a centralized approach, we adopt a distributed approach where each agent has an extended local model and decision-making process. We have introduced a pre-negotiation phase that allows agents to transfer meta-level information on related negotiation issues. Using this information, the agent can build a more accurate model of the negotiation in terms of modeling the relationship of flexibility and success probability. This more accurate model helps the agent in choosing the appropriate negotiation solution. The experimental data shows that these mechanisms improve the agent's and the system's overall performance significantly. In future extension of this work, we would like to develop mechanisms to verify how reliable the agents are. We also recognize that the current approach of applying the meta-level information is mainly heuristic, so we would like to develop a learning mechanism that enables the agent to learn how to use such information to adjust its local model from previous experience. To further verify this distributed approach, we would like to develop a centralized approach, so we can evaluate how good the solution from this distributed approach is compared to the optimal solution found by the centralized approach.

8. REFERENCES

[1] S. S. Fatima, M. Wooldridge, and N. R. Jennings. Optimal negotiation strategies for agents with incomplete information. In *Revised Papers from the 8th International Workshop on Intelligent Agents VIII*, pages 377–392. Springer-Verlag, 2002.

[2] L. Hunsberger and B. J. Grosz. A combinatorial auction for collaborative planning. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000)*, 2000.

[3] N. R. Jennings, P. Faratin, T. J. Norman, P. O'Brien, B. Odgers, and J. L. Alty. Implementing a business process management system using adept: A real-world case study. *Int. Journal of Applied Artificial Intelligence*, 2000.

[4] R. Mailler and V. Lesser. A Cooperative Mediation-Based Protocol for Dynamic, Distributed Resource Allocation. *IEEE Transaction on Systems, Man, and Cybernetics, Part C, Special Issue on Game-theoretic Analysis and Stochastic Simulation of Negotiation Agents*, 2004.

[5] T. J. Norman, A. Preece, S. Chalmers, N. R. Jennings, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, A. Gray, and N. Fiddian. Agent-based formation of virtual organisations. *Int. J. Knowledge Based Systems*, 17(2-4):103–111, 2004.

[6] T. Sandholm and V. Lesser. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS95)*, pages 328–335, 1995.

[7] J. Shen, X. Zhang, and V. Lesser. Degree of Local Cooperation and its Implication on Global Utility. *Proceedings of Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2004)*, July 2004.

[8] M. Sims, H. Mostafa, B. Horling, H. Zhang, V. Lesser, and D. Corkill. Lateral and Hierarchical Partial Centralization for Distributed Coordination and Scheduling of Complex Hierarchical Task Networks. *AAAI 2006 Spring Symposium on Distributed Plan and Schedule Management*, 2006.

[9] W. Walsh, M. Wellman, and F. Ygge. Combinatorial auctions for supply chain formation. In *Second ACM Conference on Electronic Commerce*, 2000.

[10] X. Zhang, V. Lesser, and S. Abdallah. Efficient management of multi-linked negotiation based on a formalized model. *Autonomous Agents and MultiAgent Systems*, 10(2):165–205, 2005.

[11] X. Zhang, V. Lesser, and T. Wagner. Integrative negotiation among agents situated in organizations. *IEEE Transactions on System, Man, and Cybernetics: Part C, Special Issue on Game-theoretic Analysis and Stochastic Simulation of Negotiation Agents*, 36(1):19–30, January 2006.

[12] Q. Zheng and X. Zhang. Automatic formation and analysis of multi-agent virtual organization. *Journal of the Brazilian Computer Society: Special Issue on Agents Organizations*, 11(1):74–89, July 2005.