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 agents do not cheat or intend to hurt other agents even as they focus on their own goals. 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 good negotiation solution. 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.

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. Agents need to efficiently negotiate over multiple related issues concurrently in a complex, distributed setting where there are deadlines by which when the negotiations must be completed. The need for this type of negotiation can arise from acquiring either multiple resources for a single goal or resources for multiple goals that need to be solved concurrently. 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 agents do not cheat or intend to hurt other agents even as they focus on their own goals. However, 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 [13]. An example of such a system would be a virtual organization [8, 14] (i.e. a supply chain) dynamically formed in an electronic marketplace such as the one developed by the CONOISE project [7]. The virtual organization composes a number of different semi-dependent entities in order to respond more efficiently to a set of external requests over time. To accomplish tasks continuously arriving in the virtual organization, cooperation and subtask relocation are needed and preferred. However, no single agent has authority over all other agents. Each agent has limited resources and capacity; it needs to decide what to do, when to do it and how to do it according to its own goals and performance measures. Meanwhile, the performance of each individual agent is tightly related to other agents' cooperation and the virtual organization's overall performance. If no task can be accomplished because of lack of cooperation, no agent can collect any utility. Additionally, there are multiple encounters among agents since new tasks are arriving all the time. In such negotiations, price is important but not the primary factor; other factors like quality and delivery time are important too. In such systems, being truthful is in the agent's best interest. The negotiation in such systems is **different from the purely self-interested negotiation** using game theoretic [15, 9] approaches where no cooperation is necessary and no long-term relationship among agents is assumed.

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 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:

1. How much time should be spent on each negotiation.
2. How much flexibility should be allocated for each negotiation.
3. In what order should the negotiations be performed.

The above decisions are based on the attributes of each negotiation and the relationships among different negotiations,

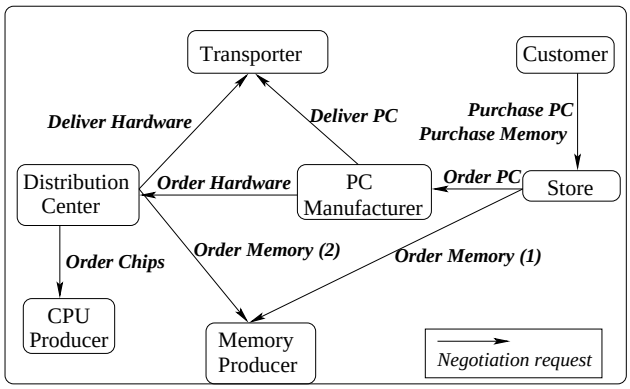

Figure 1: A Complex Supply-Chain Scenario

where the goal is to maximize the likelihood of successful negotiations by deadlines, minimize the possibility of backtracking and the decommitment penalty, so as to optimize the overall expected utility. 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.

Our previous work on multi-linked negotiation [12] described the situation where one agent needs to negotiate with multiple agents about different subjects (tasks, conflicts, or resource requirements), and the negotiation over one subject affects the negotiations over other subjects. A formalized model has been developed for multi-linked negotiation, which allows the agent to reason about the relationships among multiple related issues. Based on this model, we also developed a decision-making process for the agent to make decisions on how to order these negotiations and how to assign value for those attributes (also referred to as "features") in negotiation so as to minimize the probability of conflicts among concurrent negotiations and maximize the expected utility. These mechanisms enabled an agent to manage the multi-linked negotiations from its local perspective and choose the appropriate negotiation strategy based on the knowledge about its multiple negotiations and the interrelationships among them.

However, an even more difficult problem occurs when multi-linked negotiations are spread over multiple agents and form a negotiation chain (i.e., in Figure 1, Customer - Store - PC Manufacturer - Distribution Center - Producers - Transporters). The result of one negotiation can affect multiple other negotiations occurring at different agents. If all these negotiations are processed sequentially, it would take a very long time before a mutually agreeable solution is reached. The negative consequence of this delay could cause the customer to choose another vendor to provide the product. For example, the Store has to wait for the PC Manufacturer to finish its negotiations with the Transporter and the Distribution Center before it can reply to the customer. To reply to the request from the PC Manufacturer, the Distribution Center needs to first get replies from the CPU Producer, the Memory Producer and the Transporter, which causes additional waiting time for the customer. If all these negotiations are processed in parallel, the possible conflicts among the results of different negotiations require some issues to be renegotiated. This re-negotiation (backtracking) can then spread to other agents along the chain like a domino effect.

In this paper we extend our multi-linked negotiation model from a single-agent perspective to a multi-agent perspective, so that a group of agents involved in a chain of interrelated negotiations can find a nearly-optimal negotiation solution for pursuing their negotiations. In order to accomplish this in a distributed way, we feel it is very important for agents to have more than a local view in a negotiation chain problem, so each agent can build its local decision not only on its local knowledge but also on other agents' knowledge of the negotiation chain. As part of our approach, we introduced a *pre-negotiation* phase that allows agents to exchange meta-level information so they can adjust their local control parameters. By adjusting the local model based on this transferred meta-level information, the individual multi-linked negotiation problems are linked together, and each agent is able to incorporate some global view in their local decision-making process. In this way, these local decision-making processes are linked together through the pre-negotiation, so that the combination of each local solution becomes a good global solution.

It can be questioned whether this *pre-negotiation* really works for self-interested agents who might be lying about the transferred information. We feel that this mechanism is realistic in semi-cooperative multi-agent systems for the following reasons. First, lying is not necessarily beneficial for the agent when there are competitors. For example, if the transporter agent pretends to be extremely busy in order to gain more flexibility from the PC manufacturer, the manufacture can find other transporters. Neither is it wise to pretend to be very free, which only ends up with conflicts and failure in negotiation. Secondly, when there are multiple encounters repeated among agents, it is possible to develop mechanisms [1] to verify how reliable the agents are, which provides another incentive for agents to be truthful.

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 that is used as an example to explain related ideas. Section 4 details how to solve those problems arising in the negotiation chain. Section 5 reports on the experimental work to evaluate the effect of different negotiation policies on the agent's performance and the system's overall performance. Section 6 discusses related work and Section 7 presents conclusions and areas of future work.

2. BACKGROUND AND PREVIOUS WORK

In this work, the negotiation process between any pair of agents is based on an extended version of the contract net: 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. Details of this protocol is described in [12].

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 attributes-**in**-negotiation which are the features of the subject to be negotiated, and they are domain dependent. Another type of attributes are the attributes-**of**-negotiation, which describe the negotiation process itself and are domain independent, such as:

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

These attributes described above 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 in 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. For a more detailed explanation on this issue see [12], it also explains why concurrent negotiations is not always a good idea, and the order of negotiation is important.

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**. 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:

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

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. $P(\chi_i, \varphi)$ denotes the probability of the outcome χ_i given the feature assignment φ. $R(\chi_i, \varphi)$ denotes the agent's utility increase given the outcome χ_i and the feature assignment φ,

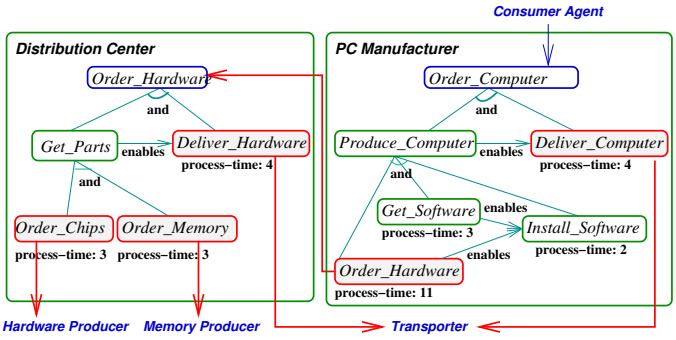

Figure 2: Task Structures of PC Manufacturer and Distribution Center

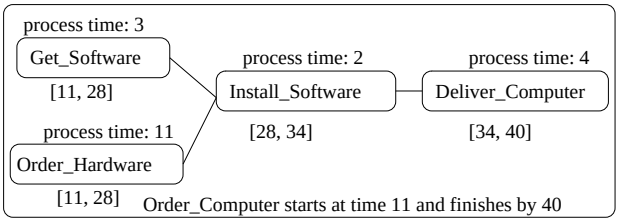

Figure 3: A Sample Local Schedule of the PC Manufacturer

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.

3. A COMPLEX SCENARIO

Figure 1 described 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.These 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 2 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 arrangements 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 3 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,

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

How should the agent manage flexibility on different tasks that require negotiation? More flexibility on one task means less flexibility on another task. Also, the flexibility on subtasks affect the finish time of the whole task. In our work on single-agent multi-linked negotiation [12], this decision is made as part of the negotiation solution found by a search procedure. 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. We use a function to describe the success probability of negotiation v: $p_s(v) = \zeta(a_1, a_2, ..., a_k)$. a_j $(j = 1, ..., k)$ represent the attributes that affect the success probability of the negotiation v. Flexibility is one of these attributes; there are other attributes, such as reward and negotiation start time, that could also affect the negotiation outcome. In this work we only focus on the flexibility. This function is domain dependent. We assume the agent has such a function available as part of its knowledge. How the agent should construct such a function, especially in a negotiation chain problem, is one of the key focuses of this work.

4. NEW MECHANISMS

We will be concerned with two features that have strong implications for the performance of a negotiation chain. The first is the *amount of flexibility* specified in the negotiation parameter. For example, if *Order Hardware* is expected to take 11 time units, the earliest start time is specified as time 40, and the deadline is specified as time 51, there is no flexibility in the outcome of the negotiation process. Either it is done starting at time 40 or it cannot be done. A more flexible negotiation structure would be one that specifies the deadline as 60; thus, the agent working on *Order Hardware* (the Distribution Center) has more freedom to find a way to accomplish this task given it may have already committed to other tasks.

The second feature we will explore is the time allocated for the negotiation process to complete. For instance, in the previous example, the negotiation on *Order Hardware* definitely should be completed by time 39, which is a hard deadline for the negotiation on *Order Hardware*. The question is, when should the negotiation on *Order Hardware* be started? It could be started once the PC manufacturer knows it needs the negotiation on *Order Hardware* (suppose at time 20), or it can be started after the PC manufacturer completes another negotiation on *Order Computer* (could be time 30). Another possible approach is to complete the negotiation on *Order Hardware* before starting the negotiation on *Order Computer*. In this case, the negotiation on *Order Hardware* needs to be completed before time 30 so there is time left for negotiation on *Order Computer*. 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

In order for an agent to take into account a more exogenous view on the negotiation characteristics, we introduce a *pre-negotiation* phase into the local negotiation process. During the pre-negotiation phase, agents communicate 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 time 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. A variable **negCount** is used to represent the number of additional negotiations needed. In a negotiation chain situation, this information is being propagated and updated through the chain until every agent has the accurate information. For example, when the PC manufacturer receives a message from the Distribution Center that contains:
  $negCount(Order\_Hardware) = 3$
  it updates its local information:
  $negCount(Order\_PC) = negCount(Order\_Hardware) + 2$
  and sends the updated information to the Store.
- Whether there are other tasks competing with this task and what is the likelihood of conflict. 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 the following formula [10]:

$$\begin{aligned} Pc_{ij} &= P(sl_i - dur_j \leq est_j - est_i \leq dur_i - sl_j) \\ &= \sum_{z=-\infty}^{+\infty} \sum_{y=z}^{+\infty} (1 - \prod_{x=z}^{y} (1 - P_{est_j - est_i}(x))) \\ &\quad \cdot P_{dur_i - sl_j}(y) P_{sl_i - dur_j}(z) \end{aligned} \quad (1)$$

  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 using the following formula: $P_{noConflict}(i) = \prod_{j=1, j\neq i}^{n} (1 - Pc_{ij})$

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 ($P_{noConflict} = 1.0$).

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 adopt the following formula for the success probability function based on the flexibility of the negotiation issue:

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

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 constant parameter used to adjust the relationship. 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 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.
$p_{bs}(v) = max(1.0, P_{noConflict}(v) * (\pi/2)/(\arctan(f(v) + c)))$

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 4 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 task *Order Memory* to increase the likelihood of success in negotiation. In the agent's further negotiation process, formula 2 with different parameter values is used in reasoning on how much flexibility should be allocated to a certain issue.

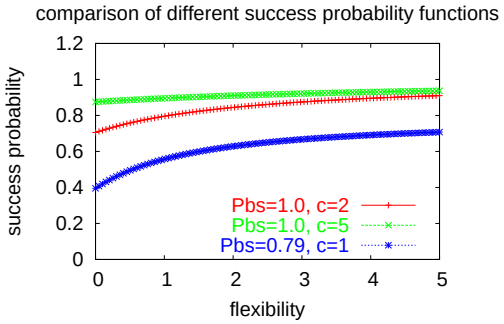

Figure 4: Different Success Probability Functions

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

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. We will discuss the effect of this mechanism in Section 5.

4.2 Negotiation Duration and Deadline

Another issue that arises when extending the multi-linked negotiation model to a negotiation chain scenario is the need to make a decision about the duration and deadline for each negotiation v. Negotiation duration $\delta(v)$ and negotiation deadline $\epsilon(v)$ are 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 deal with the negotiation which has more uncertainty (low success probability) first (if possible), so as to reduce the risk of decommitment and hence reduce the decommitment penalty.

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.

In the original model for single agent [12], 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 to reply by time 20. Time 20 can be considered as the negotiation deadline for *Purchase Computer*. When the Store 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 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. The negotiation deadline of *Order Computer* for the PC Manufacturer is actually decided based on the negotiation duration of *Order Computer* for the Store. How much time the Store 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 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.

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

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

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 the decommitment penalty. However, it is not very clear how strategy 2 and 3 perform, and we will discuss some experimental results related to this question 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 using the MASS simulator environment [4]. We performed two sets of experiments to study how the success probability functions and negotiation deadlines affect the negotiation outcome, the agents' utilities and the system's overall utility.

In the pre-negotiation phase, agents exchange meta-level information about different negotiation issues, such as whether there is further negotiation related to this negotiation ($negCount$), and if there are other tasks that are potentially competing with this task and what the likelihood of conflict ($P_{noConflict}$) is. According to this information, the local agent adjusts the parameters (P_{bs}, c) in the success probability function $p_s(v)$ to reflect how the probability of success is related to the flexibility of the task. The time needed for pre-negotiation depends on the length of the negotiation chain. Every agent updates its local information and send updated information to related agents when it receives a piece of new information from another agent.

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

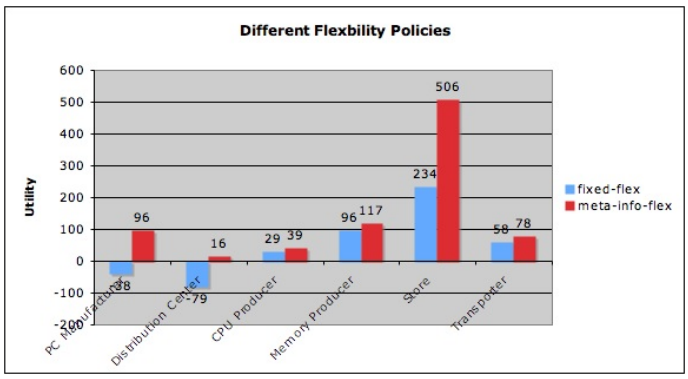

Figure 5: Different Flexibility Policies

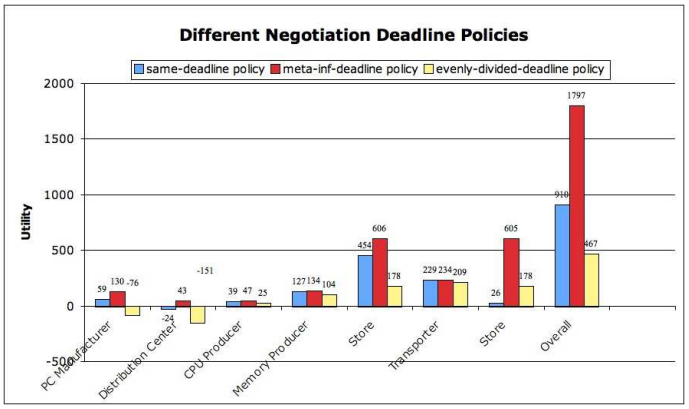

Figure 6: Different Negotiation Deadline Policies

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 5 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 phase. One *Purchase PC* task is generated every 20 time units, and two *Purchase Memory* tasks are generated every 20 time units. The deadline for task *Purchase PC* 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.

The second set of experiments studies how different negotiation deadline policies affect the performance in the negotiation chain. 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 use a different setup than the first one. We increase the number of new tasks generated to raise the average workload in the system. One *Purchase PC* task is generated every 15 time units, three *Purchase Memory* tasks are generated every 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 6). 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 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 a decommitment penalty. 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. For example, when the PC Manufacturer evenly divides the 6 time units among the two negotiations (*Produce Computer* and *Deliver Computer*), each get 3 time units. Thus, the Distribution Center must reply within 2 time units about task *Produce Computer* (1 time unit has already been spent on the communication). In our current system setting, this is an urgent request that necessitates the agent bypassing the local negotiation control process (which arranges the appropriate flexibility for each negotiation) and instead adopt a quick reply process, where no detailed reasoning on flexibility is involved. Therefore, even

if the meta-info-flexibility policy is adopted in this experiment, it may not affect the negotiation strategy since there is insufficient time for negotiation. This explains the bad performance of the evenly-divided-deadline policy in Figure 6.

6. RELATED WORK

Fatima, Wooldridge and Jennings [3] 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 [6] 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 combinatorial auction [5, 11] 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, since all items are announced at the same time, meaning all issues are negotiated concurrently. This would lead to a problem similar to those we discussed when the same-deadline policy is used. Also, a combinatorial auction is unrealistic for this problem because the range of possible bids each agent can make with respect to how it can schedule its local tasks/resources is enormous. Even though bid elicitation [2] is a possible approach to reducing the number of bids that need to be generated it does not seem feasible for this type of problem because of the complex nature of the temporal constraints in each agent.

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 agent can also use this information to allocate appropriate time for each negotiation, so as to find a good ordering of all related negotiations. 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. Additionally, we would like to develop a learning mechanism that enables the agent to learn how to adjust these values from previous experience. Also we would like to introduce some coordinators (agents who are responsible for part of the negotiation chain) and examine whether it would further facilitate the process. To further verify this distributed approach, we would like to develop a centralized approach as a base of comparison, so we can evaluate how good the solution from this distribution approach is compared to the optimal solution found by the centralized approach.

8. REFERENCES

[1] R. Ashri, S. D. Ramchurn, J. Sabater, M. Luck, and N. R. Jennings. Trust evaluation through relationship analysis. In *Proc. 4th Int Joint Conf on Autonomous Agents and Multi-Agent Systems*, Utrecht, Netherlands, 2005.

[2] W. Conen and T. Sandholm. Preference elicitation in combinatorial auctions: Extended abstract. In *ACM Conference on Electronic Commerce (ACM-EC)*, Tampa, FL, October 14-17 2001.

[3] 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.

[4] B. Horling, R. Vincent, and V. Lesser. Multi-agent system simulation framework. In *16th IMACS World Congress 2000 on Scientific Computation, Applied Mathematics and Simulation*. EPFL, August 2000.

[5] 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.

[6] 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.

[7] 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.

[8] E. Oliveira and A. P. Rocha. Agents advanced features for negotiation in electronic commerce and virtual organisations formation processes. In *AgentLink 2001*, pages 78–97. AgentLink, 2001.

[9] G. Z. Sarit Kraus, Jonathan Wilkenfeld. Multiagent negotiation under time constraints. *Artificial Intelligence*, 1995.

[10] 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.

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

[12] 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.

[13] 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*, Accepted, to appear.

[14] 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.

[15] G. Zlotkin and J. S. Rosenschein. Mechanism design for automated negotiation, and its application to task oriented domains. *Artificial Intelligence*, 1996.