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

February 29, 2012

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 agents' 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 improve the accuracy of its local model about how other agents would react to the negotiations. This more accurate model helps the agent in choosing a better negotiation solution for a distributed negotiation chain problem. 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, Semi-Cooperative Systems

1 Introduction

Effective negotiation for sophisticated 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. Multi-linked negotiation (MLN) is required when an agent needs to perform **multiple negotiations, each involving multiple other agents, and where the outcome of each negotiation influences the other negotiations**. 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. For example, in a distributed surveillance system [4], multiple sensors belonging to different organizations are needed to monitor a suspect target. Negotiations are performed to acquire these resources. The negotiation outcome - when each sensor would be made available, affects the negotiations for the other sensors given the time constraints on these data collection tasks. Another example of MLN is a computer-assisted crisis management scenario [23]. When a severe weather incident occurs, the responders from different departments need to accomplish various tasks, such as to clear the road, transport injured people to the hospital and repair the power system. To coordinate these activities, a large number of MLNs need to be performed efficiently. This is an important research area where very little work has been 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 agents' cooperation and the system's overall performance, such as, the total number of tasks being accomplished in the system. There is no single unified 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 constraints and uncertainties, and thus no agent has sufficient knowledge or computational resources to determine what is best for the whole system [27]. An example of such a system would be a virtual organization [15, 28]; for example, a supply chain dynamically formed in an electronic marketplace such as the one developed by the CONOISE project [14]. The virtual organization consists of a number of entities that respond to a set of external requests over time. To accomplish tasks continuously arriving in the virtual organization, cooperation and sub-task relocation are needed and preferred. However, no single agent has authority over all other agents. There is no single global goal since each agent may be involved in multiple virtual organizations. 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. Similar issues of coalition formation and their operation that require MLN support are also arising in the next-generation of distributed operating systems, such as the distributed version of System S for stream processing being developed at IBM Research in Hawthorne [6].

The negotiation in semi-cooperative multi-agent 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 negotiation factors like quality and delivery time are important too, and may affect the utility that is received as a result of task completion. Further long-term contract relationships among agents may make them more willing to accommodate the requests of others if there is no significant negative influence on its local utility. Another effect of repeated interactions among agents is that reputation mechanisms in the system become viable as a way of making cheating unattractive from a long-term viewpoint [21]. Furthermore, it is difficult to cheat effectively in such systems, given the incomplete/uncertain information and the dynamic environment. To summarize, this work focuses on systems where agents are **self-interested** because they primarily focus on their own goals; but they are also **semi-cooperative**, meaning that with the existence of long-term contracts and reputation mechanisms, they are willing to be truthful and collaborate with other agents to find solutions that are beneficial to all participants, including themselves. In other word, they are looking for balance between their short-term rewards with their long-term interest. To maximize one's utility in a single negotiation session with strategic bargaining mechanism is not the most important goal for semi-cooperative agents, they are more interested in finding feasible and efficient outcomes for multiple interrelated negotiations. Self-interested agents can be semi-cooperative when their own utility tightly relates to other agents' utilities in the system. On the other hand, agents in a cooperative system can also be semi-cooperative due to multiple current goals, limited communication and computation resources, and therefore no agent has the global view or complete knowledge to decide what is best for the system in this situation; individual agents may be better off focusing on their local goals [27]. As part of the experimental work presented in this paper, we will also show that the mechanisms developed here all also appropriate for cooperative systems.

The goal of this research is to develop a set of **macro-strategies** that allow the agents to effectively manage multi-linked negotiations, including, but not limited to the following decisions:

1. How much time should be spent on each negotiation? The more time is spent on negotiation, the less time is available for task execution. The more time spent on one negotiation means less time for another negotiation if these two negotiations need to be performed in sequence.
2. How much *flexibility* (see formal definition in Formula 5) should be allocated for each task in negotiation? This defines a range between the earliest start time and the deadline for the task to be completed.

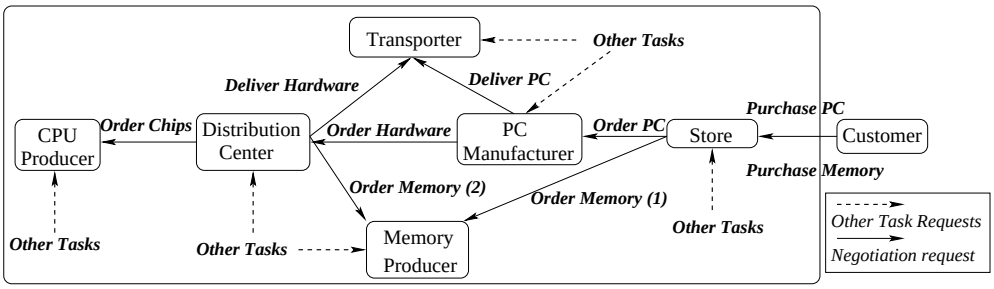

Figure 1: A Complex negotiation-chain Scenario

3. In what order should the negotiations be performed? This specifies a particular ordering to perform the multiple related negotiations.

The above decisions are based on the attributes of each negotiation and the relationships among their different negotiations that the agent is involved in, where the goal is to increase the likelihood of successful negotiations by deadlines, decrease the possibility of decommitting from previously made commitments (associated with 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.[10]. A 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. This work on negotiation is concerned more about the meta-level decision-making process in negotiation rather than the basic protocols or languages. The negotiation chain problem studied in this paper is different from the SCM game in the trading agent competition [1], where both Consumers and Suppliers are programmed with pre-determined policies. The agent is solving a local optimal problem focusing on planning, scheduling and coordination with a relatively static global context, it does not need to reason about the ordering and deadlines of multiple related negotiations.

Our previous work on multi-linked negotiation [25] 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 in the same agent. A model we developed for multi-linked negotiation allows the agent to reason about the relationships among multiple related negotiations occurring in that agent. Based on this model, we also developed a meta-level 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 enable an agent to manage the multi-linked negotiations from its local perspective and choose the appropriate negotiation strategy based on 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 distributed negotiation chain (i.e., in Figure 1, Customer - Store - PC Manufacturer - Distribution Center - Producers - Transporters). These agents need to negotiate about the quantity, quality and delivery time of the products/parts, and the start time and deadline of the transportation tasks. They are also negotiating on the reward/payment and decommitment penalty if the commitment is broken. 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, in Figure 1, the Store Agent 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 could require some issues to be renegotiated. For example, the PC Manufacturer has to re-negotiate with the Store about the delivery time after it discovers that the Transporter cannot deliver the computers to the store as it promised in previous negotiation. This re-negotiation (backtracking) can then spread to other agents along the chain like a domino effect [1].

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 chains of interrelated negotiations can find nearly-optimal macro negotiation strategies for pursuing their negotiations. In order to accomplish this in a distributed way, we feel it is very important for agents to have a broad view of the negotiation-chain problem, so each agent can build its local decision based not only on its local knowledge but also on other

[1] Alternatively, re-negotiation can be replaced by accepting de-commitment penalties (again with the consequence to rearrange any affected issues by this de-commitment). In our experimental work described later, we adopt this approach - when an agent finds that it cannot fulfill the commitment it made before, it drops the task, informs the other agent and pays the de-commitment penalty.

agents' knowledge of the negotiation chain. We see this exchange as feasible in semi-cooperative or fully cooperative multi-agent systems. As part of this extension, we introduce 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 non-local view in their local decision-making process [20]. Therefore, these local decision-making processes are better informed through the pre-negotiation process, and the combination of local macro negotiation strategies is more likely representing a good combined macro negotiation strategy from a global perspective. Based on the transferred meta-level information, the agent can reason on how to allocate appropriate time and flexibility for each related negotiation. This approach is a completely distributed approach; there is no centralized authority or mediator in the system and each agent has full control of its local activities. In addition only a small amount of meta-information needs to be transferred. This meta-information is the abstract description of their problem-solving states, and the agents only transfer the information before the first negotiation begins or when there is significant change in the environment.

The remainder of this paper is structured in the following manner. Section 2 describes the basic negotiation process and briefly reviews our previous work on 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 distributed 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 reports the comparison of this distributed approach with a centralized approach. The generality of this approach is discussed in Section 7. Section 8 discusses related work and Section 9 presents conclusions and areas of future work.

2 Basic Negotiation Process and Previous Work

In this work, the negotiation process between any pair of agents is based on an extended version of the contract net [18]: the initiator agent announces the proposal including multiple features; the responding agents evaluate it and respond 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 as it has promised, it needs to pay the other party a decommitment penalty as specified in the commitment. Details of this protocol are described in [25].

A negotiation starts with a proposal that announces that a task (t) needs to be performed. The proposal 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. *regular reward* (r): if the task is finished as the contract requested, the contractor agent will get reward r.
4. *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.
5. *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. Depending on the actual application, some of these attributes may be removed and others may be added to better specify the negotiation subject in a particular domain. Another type of attribute 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.).

These attributes described above are similar to those used in project management; however, **the multi-linked negotiation problem cannot be reduced to a traditional 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

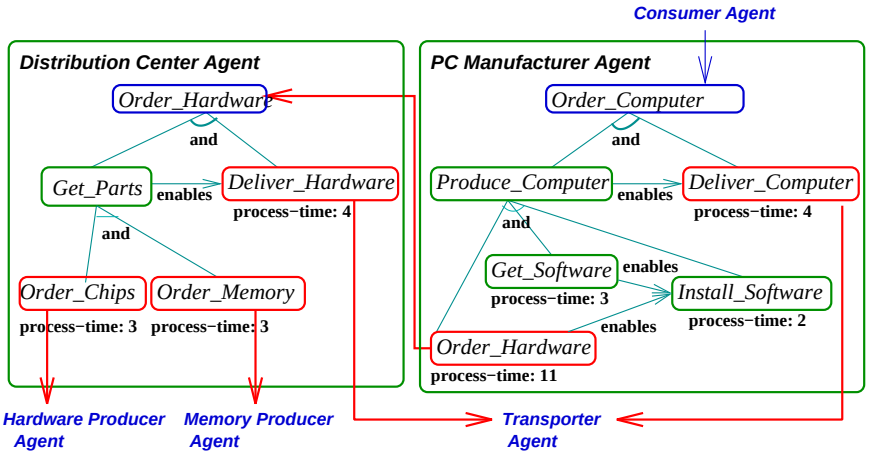

Figure 2: Task Structures of PC Manufacturer Agent and Distribution Center Agent

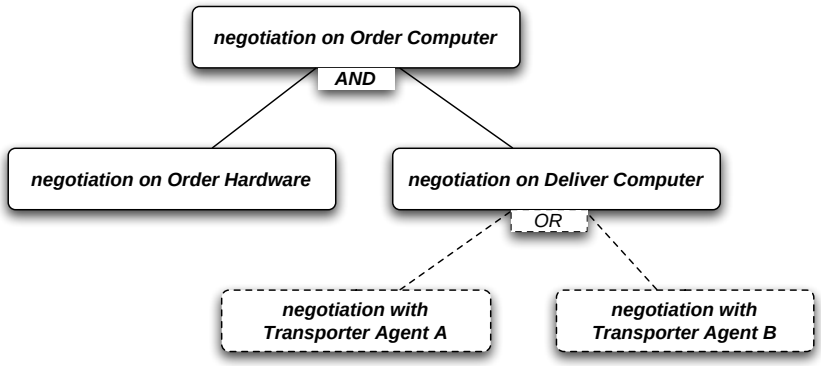

Figure 3: The multi-linked negotiation problem for the PC Manufacturer Agent

of negotiations and the attributes of the subjects are interrelated as well. This two-dimensional complexity of interrelationships distinguishes it from a classic scheduling problem, where all tasks to be scheduled are local tasks and no negotiation is needed. For a more detailed explanation on this issue see [25], which also explains why concurrent negotiation is not always a good idea, and why 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**. We developed a formal model of the multi-linked negotiation problem from a single agent's perspective [25], presented below.

Definition 2.1 *A* **multi-linked negotiation problem** *is defined as an undirected graph (more specifically, a forest as a set of rooted trees):* $M = (V, E)$*, 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 that all its children nodes have successful accomplishments. The following example illustrates this idea. Figure 2 shows the local task structures of the PC Manufacturer Agent and the Distribution Center. For the PC Manufacturer Agent, the negotiation on task *Order Computer* node has two children, negotiation on task *Order Hardware* and negotiation on task *Deliver Computer*, each representing a subtask that cannot be done locally. These negotiations are related because how and when task *Order Hardware* and task *Deliver Computer* are performed directly affect task *Order Computer*. The multi-linked negotiation problem of PC Manufacturer Agent is shown in Figure 3. The successful accomplishment of the commitment on task *Order Computer* depends on the successful accomplishment of both task *Order Hardware* and task *Deliver Computer*. On the other hand, the OR relationship associated with a negotiation v means the successful accomplishment of the commitment on v requires that at least one child node has successful accomplishment, where the multiple children nodes represent alternatives to accomplishing the same goal, such as to negotiate the same issue with different potential contractors. For instance, if there are two transporter agents A and B both can

potentially deliver the computer, two nodes *negotiation with Transporter Agent A* and *negotiation with Transporter Agent B* can be added as the children for the node *negotiation on deliver computer* in Figure 2, with an OR relationship associated with it.

Given these interrelationships among negotiations and the decommitment penalty in negotiation, the ordering of the negotiations becomes important because it affects agents' utilities. Assume that the PC Manufacturer Agent first negotiates with the Consumer Agent on task *Order Computer* and finds a mutually-agreed commitment. Later in the negotiation on task *Order Hardware* and task *Deliver Computer*, it finds that it cannot fulfill the commitment on task *Order Computer* because some attributes in the later negotiations do not support the earlier commitment, such as the finish time of the task. In this case, the PC Manufacturer Agent needs to pay a decommitment penalty to the Consumer Agent. The optimal negotiation ordering of all related negotiations should maximize the expected utility, which is the the expected reward minus the expected decommitment penalty. The evaluation of a negotiation ordering is based on the following issues: the probability of each negotiation succeeding, and the decommitment penalty associated with each negotiation if the commitment has to be revoked after other related negotiations fail. Additionally, the negotiation ordering also affects the probability of a negotiation succeeding because the start time of a negotiation affects the outcome of a negotiation in general, i.e. a later negotiation start time limits the number of options that can be explored in the negotiation process in finding a mutually agreeable contract; while an earlier negotiation start time increases the likelihood of decommitting given that other related negotiation results are yet unknown and they may be inconsistant with this negotiation outcome.

Each negotiation $v_i (v_i \in V)$ is associated with a set of attributes $\mathcal{A}_i = \{a_{ij}\}$. Each attribute a_{ij} either has already been determined or needs to be decided. How the values for those features (referred as *feature assignment*) are selected is important because they affect the negotiation outcome and hence affect the agent's overall utility. In our previous work on a single agent's multiple related negotiations, we dealt with the feature assignment problem in the following way (we use the example in Figure 2 in the following discussion, and the problem is analyzed from the PC Manufacturer Agent's viewpoint):

1. For attributes-**in**-negotiation, there are two possibilities. For incoming task requirement, i.e. the task *Order Computer*, the values of most of these attributes, such as earliest start time, deadline, regular reward, etc. have already been determined by the Consumer Agent when it proposes this task. The PC Manufacturer Agent needs to decide whether to accept this proposal and if so, determine the promised finish time. It may receive some extra early finish reward, depending on the early finish reward rate in the proposal. For outgoing task requirements, such as task *Order Hardware* and task *Deliver Computer*, the values of these attributes **in** negotiation are not pre-determined. The PC Manufacturer Agent needs to find out the values for these attributes. In the negotiation context, the values are not single values, but a range of acceptable values. For example, if the deadline of task *Order Hardware* has a time range of $[10, 15]$, it means that any time that falls within this range is acceptable; in other words, there exists a feasible local schedule for the PC Manufacturer Agent to fulfill the commitment on task *Order Computer*, assuming the negotiation on task *Deliver Computer* reaches an agreement within the assigned time ranges for its attributes. Such time ranges are also referred to as *consistent ranges*; finding such ranges allows negotiations to be performed simultaneously. We developed a set of partial-order reasoning tools to find all possible consistent ranges for temporal attributes, and use these possible candidates as the search space for the best overall solution in terms of the sequencing of negotiation and their attributes. The search algorithm is described later.

2. For attributes-**of**-negotiation, it is more complicated because most of them are not pre-determined. The negotiation start time depends on the negotiation ordering and negotiation durations of other negotiations. The negotiation durations and negotiation deadlines affect the decision on negotiation ordering. A negotiation ordering is **valid** if all negotiations can be finished before their deadlines. The negotiation deadline is determined by the agent who initializes the negotiation. Negotiation duration represents how much time the agent wants to spend on this negotiation. For example, if the PC Manufacturer Agent decides that the negotiation duration for task *Order Hardware* is 5, given the negotiation ordering it selects, the start time for this negotiation is 10, which means the negotiation deadline for this task is 15. If an agreement cannot be found before time 15, this negotiation is considered a failure. For all outgoing tasks, the agent needs to decide how much time it should spend on each negotiation (the negotiation duration). This decision affects the possible negotiation ordering and also affects the negotiation outcome. In our previous work, we did not develop a strategy on how to make this decision; we simply assume a fixed duration for all negotiations. However, we cannot use such a simplified assumption when we extend this model to a negotiation-chain scenario. We solve this problem in Section 4. The success probability of negotiation is another important attribute, which is used in the evaluation of a negotiation solution. The success probability is modeled as a function of other attributes, the more accurate this model, the better the agent's performance. The second component of this work is to use meta-level information from other agents to construct a more accurate model of this function in a negotiation-chain scenario.

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:

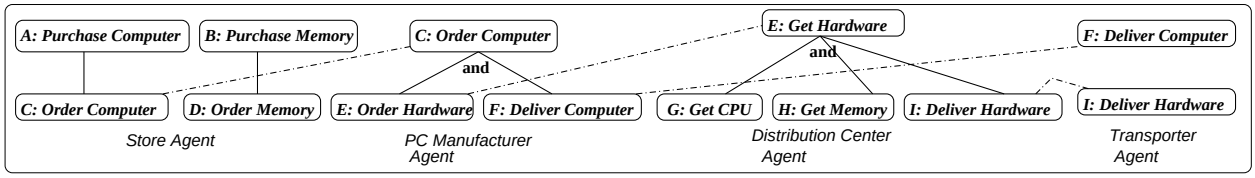

Figure 4: Distributed Model of negotiation chains

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

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[2], and $C(\chi_i, \phi, \varphi)$ is the sum of the decommitment penalties of those negotiations, which are successful, but need to be abandoned due to 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.

We have developed a heuristic search algorithm [25] to solve the single agent's multi-linked negotiation problem that produces nearly-optimal solutions. The algorithm is briefly described as follows:

1. Find all the possible (consistent) feature assignments for all the attributes; partial order reasoning tools are used to check the consistence of temporal attributes.
2. Perform a simulated annealing search starting with an initial negotiation ordering ϕ.
3. Find the best feature assignment φ_ϕ for the given negotiation ordering ϕ, based on the evaluation of expected utility $\mathcal{EU}(\phi, \varphi_\phi)$ defined in Equation (1).
4. Modify the current negotiation ordering ϕ and get a new negotiation ordering ϕ_{new}, find the best feature assignment $\varphi_{\phi_{new}}$ for this negotiation ordering ϕ_{new}.
5. If $\mathcal{EU}(\phi, \varphi_\phi) < \mathcal{EU}(\phi_{new}, \varphi_{\phi_{new}})$, $\phi \leftarrow \phi_{new}$; otherwise replace ϕ with ϕ_{new} with a probability less than 1. Return to Step 4;
6. The algorithm stops after the number of repetitions has reached a pre-determined limit.

Consider an example with three negotiations A, B and C; assuming that the negotiation start time $\tau = 0$, and the negotiation duration of each negotiation is the same $\delta(v_i) = 5$. A POR represents a partial order relationship between two scheduling elements. The negotiation schedule with no POR is: $A[0, 5]B[0, 5]C[0, 5]$; the negotiation schedule with one POR ($A \rightarrow B$): $A[0, 5]B[5, 10]C[0, 5]$; the negotiation schedule with two PORs ($A \rightarrow B$, $A \rightarrow C$) is: $A[0, 5]B[5, 10]C[5, 10]$. There are also some other possible schedules. The algorithm outputs the schedule with the best value from all the possible schedules it has explored.

We still use the above search algorithm as the core of the decision-making for each individual agent in the negotiation-chain scenario, but introducing new mechanisms for determining the values of some of the parameters, i.e. negotiation deadlines and success probabilities, to reflect that this negotiation is part of a negotiation chain. 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. When an agent uses the decision-making process to choose its local negotiation decision, it needs to model how other agents would react to its particular negotiation request. The more accurate this model is, the higher the actual quality the solution gets. We propose to use meta-level information to refine the agent's local model about other agents, with the focus on the following two aspects: how to use the meta-level information to construct a better model of success probability function $P(\chi_i, \varphi)$ and how to make decisions on negotiation duration and deadline. The experimental work demonstrates that these extensions improve significantly the performance of the agents and the system where there are negotiation chains. Table 1 summarizes the parameters used in this paper.

3 Negotiation-Chain Problem

Figure 1 describes a complex negotiation-chain scenario. The customer purchases computers and memory chips from the Store Agent. The Store orders computers from the PC Manufacturer, and also orders memory chips from the Memory Producer. In order

[2]The values assigned to some features may affect the reward, i.e., the finish time affects the total reward if there is additional early reward for finishing the task earlier than the requested deadline.

Table 1: Parameters in Multi-linked Negotiations

name	symbol	value determined by / affected by
attributes-*in*-negotiation (about task)		
earliest start time	est	Their values are mostly determined by the initiator agent. The responding agent may negotiate with the initiator agent about the exact value within a range that is determined by the initiator agent. For initiator agent, these values are output. For responding agent, these values are input but may be flexible and can be changed during negotiation.
deadline	dl	
regular reward	r	
early finish reward rate	e	
decommitment penalty rate	p	
attributes-*of*-negotiation (about negotiation)		
negotiation duration	$\delta(v)$	Their values need to be determined by the initiator agent. For initiator agent, these values are output. For responding agent, these values are input.
negotiation start time	$\alpha(v)$	
negotiation deadline	$\epsilon(v)$	
success probability	$p_s(v)$	affected by multiple factors including those listed above and p_{bs} listed below.
other internal parameters		
negotiation ordering	ϕ	each agent determines the ordering of its multi-linked negotiations.
feature assignment	φ	each agent determines the values for its output attributes.
expected utility	$\mathcal{EU}(\phi, \varphi_\phi)$	depends on ϕ and φ.
negotiation count	negCount	mate-level information collected during pre-negotiation.
likelihood of conflict	Pc_{ij}	depends on the parameters of tasks with type i and type j.
likelihood of no conflict	$P_{noConflict}$	depends on Pc_{ij}.
flexibility of task t	$f(t)$	depends on est, dl and $process\_time$ of task t.
basic success probability	p_{bs}	depends on $P_{noConflict}$ and $f(t_v)$.

to fulfill the order of the Store Agent, the PC Manufacturer needs to order hardware from the Distribution Center and also ask the Transporter agent to deliver the PC to the Store Agent. Meanwhile, in order to fulfill the order from the PC Manufacturer, the Distribution Center needs to order CPU chips from the CPU Producer and order memory chips from the Memory Producer. It also needs the Transporter agent to deliver the hardware to the PC Manufacturer. All these negotiations are interrelated and they affect one another either directly or indirectly. The Store, the PC Manufacturer, the Memory Producer and the Distribution Center are all involved in multi-linked negotiation problems. Figure 4 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 the previous section. 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 4 represents the connection of these local optimization problems through the negotiation subjects they have in common.

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

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

Agent A_i's **local optimal** solution S_i^{lo} maximizes the expected local utility based on incomplete information and imperfect assumptions about other agents' local strategies. We defined such incomplete information and imperfect assumptions of agent i as I_i. Such information is an estimation of how other agents would respond to a negotiation request, i.e. there is a 30% chance that agent j will accept a task request within the next 10 hours. In other words, I_i represents the estimation of other agents' local strategies: $\{S_1^{lo}, ..., S_{i-1}^{lo}, S_{i+1}^{lo}, ..., S_n^{lo}\}$:

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

$U_i^{exp}(S_i^{lo}, I_i)$ is the expected utility that agent i can achieve when it uses local strategy S_i^{lo} given the assumption of other agents' local strategies as I_i. This expected utility may be different from the actual utility that agent i achieves: $U_i(\{S_1^{lo}, S_2^{lo}, ...S_n^{lo}\})$, when it uses local strategy S_i^{lo} and other agents use local strategies $\{S_1^{lo}, ..., S_{i-1}^{lo}, S_{i+1}^{lo}, ..., S_n^{lo}\}$ respectively. How different they are depends on how close the estimation I_i is to the other agents' local optimal strategies: $\{S_1^{lo}, ..., S_{i-1}^{lo}, S_{i+1}^{lo}, ..., S_n^{lo}\}$. If there is no difference between I_i and the other agents' local optimal strategies: $\{S_1^{lo}, ..., S_{i-1}^{lo}, S_{i+1}^{lo}, ..., S_n^{lo}\}$, then the set of local optimal strategies $\{S_1^{lo}, S_2^{lo}, ..., S_n^{lo}\}$ represents a Nash Equilibrium, meaning no agent can do better by using its local strategy S_i^{lo} while other agents use local strategies $\{S_1^{lo}, ..., S_{i-1}^{lo}, S_{i+1}^{lo}, ..., S_n^{lo}\}$. This is based on the assumptions that every agent has perfect knowledge, and also S_i^{lo} can be found, which may not be possible given each S_i^{lo} $(1 \leq i \leq n)$ is calculated depending on S_j^{lo} $(1 \leq j \leq n, j \neq i)$.

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 at least one agent's local utility is improved without any other 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**:

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

There can be 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 that the agents are making decisions concurrently. Furthermore, in a dynamic environment, it is a very difficult and sometimes even impossible task to construct perfect local information and assumptions about other agents (I_i) .

Therefore, 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 becomes more accurate, the agent's local optimal solution to its local multi-linked negotiation problem becomes a better local optimal solution in the context of the global negotiation-chain problem. We are not arguing that this statement is a universally valid statement that holds in all situations, but our experimental work shows that it is true in the environments we have tested. Our experimental results show 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 build a better function for allocating 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 task 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 frequency, the slack time and the duration, etc. When such information is not available directly, agents can learn such information from their past experience as we did in our experiments. We assume that 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 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 decisions on these subtasks; hence, the $negCount$ for each of these subtasks is 0. The following information is sent to the PC Manufacturer agent by the Distribution Center agent:
  $negCount(Order\_Hardware) = 3$
  This information does not mean that the Distribution Center agent will handle the negotiations on the three subtasks sequentially, it just implies that there are three further related negotiations existing for task *Order Hardware*. Given this information and other information, the PC Manufacturer agent will decide how much time to allocate on the negotiation for task *Order Hardware*, and by then the Distribution Center agent will decide how to handle the negotiations on the three subtasks.

  For the PC Manufacturer task *Order PC* contains two subtasks that require additional negotiations: *Deliver PC* and *Order Hardware*. When the PC Manufacturer receives the message from the Distribution Center, it updates its local information:
  $negCount(Order\_PC) = 2 + negCount(Deliver\_PC)(0) + negCount(Order\_Hardware)(3) = 5$
  and sends the updated information to the Store Agent.

  In order to calculate $negCount$, an agent needs to wait for replies from all its related agents, who need replies from their related agents too. This process goes further to the every end of each negotiation chain, and it stops if there is no cycle in the multi-linked negotiation graph. In the implementation of the pre-negotiation process, considering the possibility of communication

failure in reality, a time limit is set for waiting replies from other agents. This limit is chosen based on the estimation of the length of the longest negotiation chain in the current problem. This limit can be adjusted when there is a reply message received after the waiting period; hence a more accurate result can be achieved in the next pre-negotiation phase.

- Whether there are other tasks competing with this task and what the likelihood of conflict is. Conflict means that given all constraints, 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 the following formula [19]:

$$\begin{aligned} Pc_{ij} &= P(dl_i - est_j \leq dur_i + dur_j \wedge dl_j - est_i \leq dur_i + dur_j) \\ &= P(sl_i - dur_j \leq est_j - est_i \leq dur_i - sl_j) \\ &= \sum_{z=-\infty}^{+\infty} \sum_{y=z}^{+\infty} \sum_{x=z}^{y} P_{est_j - est_i}(x) P_{dur_i - sl_j}(y) P_{sl_i - dur_j}(z). \end{aligned} \quad (3)$$

  [28] shows that the calculation result using this formula is reasonably close to the actual simulation result.

  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 [3]:

$$P_{noConflict}(i) = \prod_{j=1, j\neq i}^{n} (1 - Pc_{ij}). \quad (4)$$

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[4]. 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 start-up period 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. An alternative way to find $P_{noConflict}$ is through direct learning based on the tightness of the agent's current schedule and the estimated arrival time and duration of the future tasks. Agents are constantly monitoring these characteristics. An updated message will be sent to related agents when there is significant change of the meta-level information.

In the above discussion, it is assumed that there is a statistical model about the characteristics of the incoming tasks, so that the agent can learn this model through its experience. However, such a statistical model may not exist in some real-world applications or there is only single-shot interaction between agents and it is impossible for agents to learn such a model. In this type of situation, the detailed calculation such as Formula 3 cannot be used but the pre-negotiation phase can still be advantageous when other types of meta-level information is transferred, such as the flexibility of current schedule, the finish time of the latest task, the size of the biggest slack window in current schedule, etc. Such information can also be used by other agents to adjust their local models about how agent A_i would respond in future negotiations, though we did not implement this in our experiments.

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 manufacturer can find other transporters. Neither is it wise to pretend to be very free, which only results in conflicts and failure in negotiation. Secondly, when there are multiple encounters repeated among agents, it is possible to develop mechanisms [3] to verify how reliable the agents are, which provides another incentive for agents to be truthful. Finally, if the agents in the chain are in a virtual enterprise such as in the CONOISE project [14], they have an incentive for the enterprise to be successful since their reward will be dependent on the success of the enterprise. In our scenario, each agent is motivated to provide local information for other agents so as to maximize the success probability of the negotiation, since the agent only collects reward when it fulfills a contract resulting from a successful negotiation.

[3] Given the experiment setting we used, the probability that there is a conflict with three tasks but no pair-wise conflict for any two tasks out of the three, is very low, so it is ignored in this formula. When such probability is high enough, this formula needs to be revised to include this situation.

[4] There are multiple *Order Chips* tasks arriving at different times, however, there is no conflict among them given they are spaced out in time dimension in the current experimental setting.

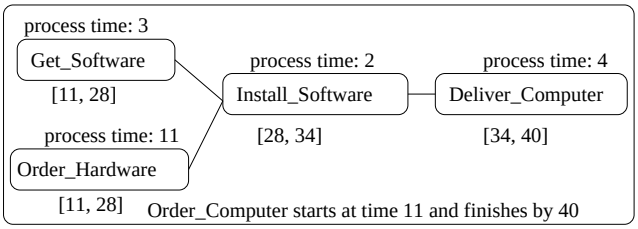

Figure 5: A Sample Local Schedule of the PC Manufacturer

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 by affecting the values of some features that are used in the agent's local decision-making process. 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 significantly affect 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 started 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. This feature is called *negotiation duration*, which is determined by the agent who initiates the negotiation session. A *deadline for replying* is set in the initial proposal; the negotiation outcome is considered a failure if no commitment is reached by this deadline. The initiator agent needs to decide how long the *negotiation duration* is for a particular negotiation session. 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). These two decisions would result in different durations allowed for the negotiation on *Order Hardware* - 19 (39-20) vs. 9 (39-30), which would affect the negotiation outcomes. 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[5]. 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 2 shows the local task structures of the PC Manufacturer and the Distribution Center Agent. 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 non-local tasks, 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 that 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 5 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 another agent to find a

[5]We recognize that the importance of efficient managing related negotiations is driven by the tight deadline and the need for urgent responses to tasks; in other words, the negotiation duration and the task execution duration have similar magnitudes. This is not the case for traditional supply chain application where production tasks take hours or days but electronic negotiation may only take seconds. However, for most cyberspace application such as dynamic target tracking with distributed sensor networks, the execution time for some tasks are at the similar magnitude as the negotiation duration.

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, the flexibility of task t is defined as $f(t)$:

$$f(t) = \frac{dl - est - process\_time(t)}{process\_time(t)}. \tag{5}$$

Flexibility of a task is an important attribute in negotiation, because it directly affects the possible outcome of the negotiation on this task. The success probability of a negotiation v on task t_v 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 task t_v in this negotiation v:

$$p_s(v) = p_{bs}(v) * (2/\pi) * (\arctan(f(t_v) + c))). \tag{6}$$

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. This is also consistent with the usual observation in human negotiations for real world problems: the chance of successfully scheduling a job increases when the flexibility of this job increases to a certain point, additional flexibility does not has significant impact afterwards. The success probability function described here in Equation (6) is only one of many possible forms. The most appropriate form of this function should be decided upon the knowledge of the application domain and individual agents scheduling mechanism and negotiation strategy, if they are available. Learning from previous interaction experience is an alternative to build a more accurate form of this function, when other agents private information is not available.

p_{bs} is the *basic success probability* of this negotiation v when the flexibility $f(t_v)$ 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 6. This function describes the agent's assumption about how the other agent involved in this negotiation would respond to this particular negotiation request, when it has flexibility $f(t_v)$. This function is part of the agent's local model about other agents' situation in this negotiation chain. 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 t_v for common resources. If so, more flexibility is needed for task t_v 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 is successful. If no further negotiations are 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 $P_{noConflict}$, $f(t_v)$ (the flexibility of task t_v in previous negotiation), and c, using the reverse format of Equation 6.

$$p_{bs}(v) = max(1.0, P_{noConflict}(t_v) * (\pi/2)/(\arctan(f(t_v) + c))) \tag{7}$$

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 6 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 6 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,

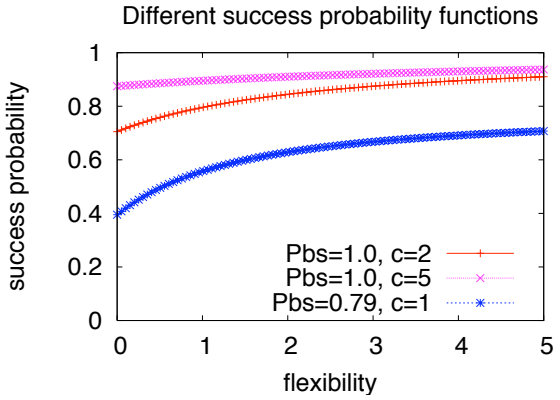

Figure 6: Different Success Probability Functions

Table 2: 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 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, the short message length and the limited number of messages 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 $\epsilon(v)$. These 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 reduce the risk of decommitment and hence reduce the decommitment penalty. A search algorithm has been developed to find such negotiation order described in [25].

For example, Table 2 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 the highest penalty (1: Order hardware) is performed after all related negotiations (2, 3, and 4) have been completed so as to reduce the possibility and cost 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 must 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 a single agent [25], 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 should the negotiation on *Order Computer* take 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

Table 3: Parameter Values Without/With Meta-level Information

	local-info-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

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: one agent's negotiation decision on how much time it should spend on each negotiation, actually affects other agents' negotiation decisions. The original model does not handle this question since it assumes the negotiation duration $\delta(v)$ is known. We propose to use the meta-level information to help deciding negotiation duration for each negotiation (*meta-info-deadline policy*). To evaluate how well it works, we compare this policy with two other straightforward approaches: *same-deadline policy* and *evenly-divided-deadline policy*.

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. During this process, the agent can reason on all related negotiations and choose a best local solution $\{S_i^{lo}\}$ to handle these multiple related negotiations. 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.

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 [8]. 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 how the success probability functions and negotiation deadlines affect the negotiation outcome, the agents' utilities and the system's overall utility. Each agent's utility is the sum of the regular reward and the early reward it receives from all its finished tasks minus the subcontract payments to other agents and the penalty it pays for canceling any commitments. The system's overall utility is the sum of each individual agent's utility. The negotiation protocol used in this work is a two-step proposal and counter proposal process based on the contract net protocol. Once an agreement is reached and one agent cannot fulfill the commitment, it needs to pay the other party a decommitment penally as specified in the commitment. Details of this protocol are described in [25]. 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 a task of a specified type is fixed and not under negotiation. Here we only describe how agents handle the negotiation duration and negotiation deadlines because these two attributes are 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), how they affect the negotiation success probability (time and cost related attributes) and how they affect the expected utility. The search algorithm and a set of partial order scheduling algorithms are used to handle these attributes.

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 sent updated information to related agents when it receives a piece of new information from another agent.

5.1 Experiment With Different Flexibility Policies

The first set of experiments is to explore the performance of different flexibility policies, which guide how agents allocate flexibilities among multiple related negotiation issues. We tried two different flexibility policies.

1. *local-info-flexibility policy*: the agent models the success probability as $p_s(v) = p_{bs}(v)$, the value of $p_{bs}(v)$ is determined 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 3 shows the values of those parameters for some negotiations.

Figure 7 shows the results of this experiment. This set of experiments includes 22 system runs, and each run is for 1000 simulating time units. In the first 200 time units, agents are learning about the task characteristics such as the distribution of the frequency, the slack time and the duration using a basic inductive learning algorithm. These task characteristics 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 were collected over the 800 time units after the pre-negotiation phase.[6] 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 (each chain is for one *Purchase Memory* task) :
Customer - Store - Memory Producer
This demonstrates that this mechanism is capable of handling multiple concurrent negotiation chains.

The results in Figure 7 are the averages of the 22 system runs. TTest result, with value in the range from 5E-20 to 5E-16, strongly supports that there is a statistical significant different between the results using the two different flexibility policies. 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.

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

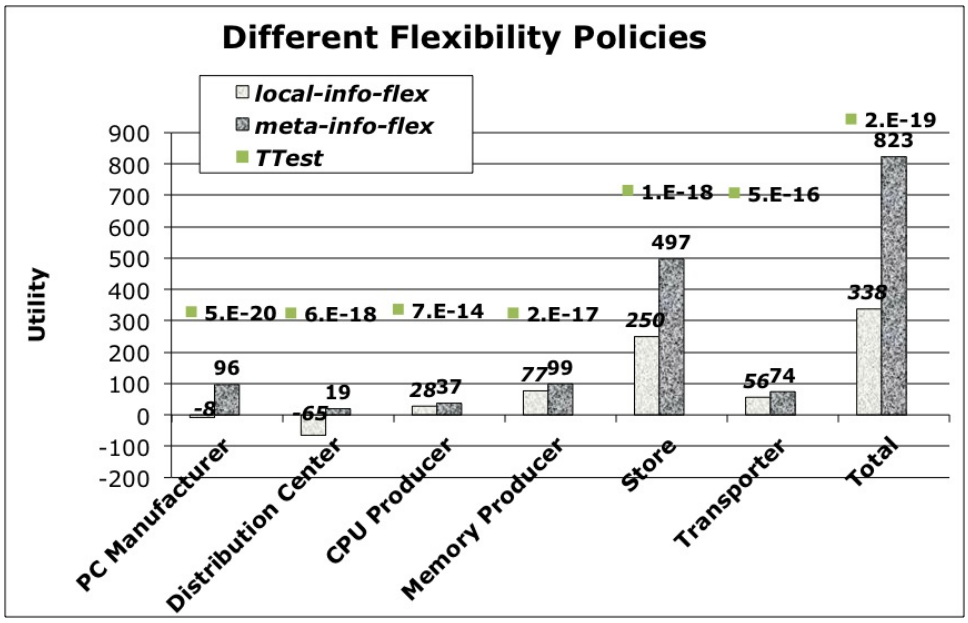

Figure 7: Different Flexibility Policies

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), more reward is received and less decommitment penalty is paid, resulting in both the agent and the system achieving better performance.

5.2 Experiment With Different Negotiation Deadline Policies

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 (using the same task frequencies as described in Section 5.1, not presented here) 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.

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 8). The results in Figure 8 are the averages of the 22 system runs. TTest was performed between the performance using the same-deadline policy and the performance using the meta-info-deadline policy, it is shown that there is a statistical significant different between the results using these two different negotiation deadline policies. 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 cancelled, 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. 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

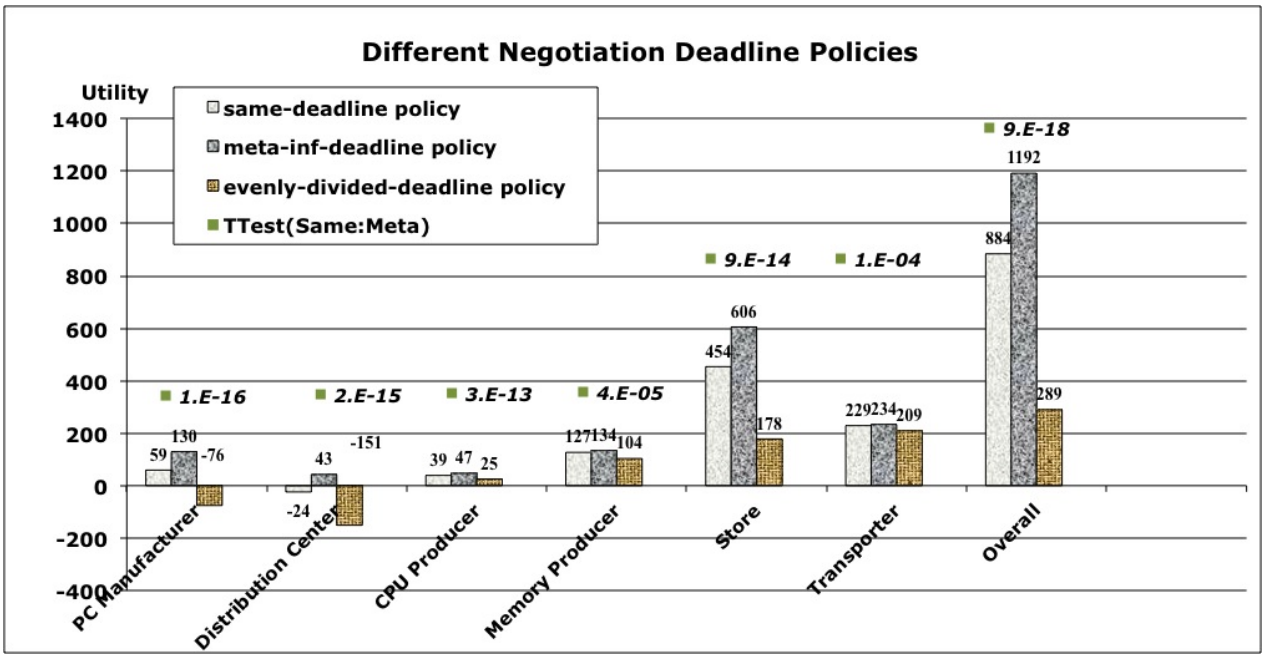

Figure 8: Different Negotiation Deadline Policies

each negotiation) and instead adopts a quick reply process, where no detailed reasoning on flexibility is involved. Therefore, even if the meta-info-flexibility policy is used 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 8.

From the above experiment results, we conclude 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 such 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 (like in our experiment setup); the efficient negotiation is especially important in such environments.

6 Comparison With A Centralized Approach

To further evaluate the performance of our distributed negotiation approaches, we compare the experimental results in Section 5 with the total utility achieved by all agents using a centralized scheduler. In this centralized approach, a centralized scheduler is used to decide which tasks are to be performed, and when, for all agents in the system. It is assumed that the information about what tasks are available and who can perform them is available when the scheduling is performed. This centralized scheduler used in this experiment is developed by the Global InfoTek Inc., as part of the DARPA Coordinator project effort. This centralized scheduler is based on the Mixed Integer Linear Programming (MILP) approach, and it schedules tasks for multiple agents with the goal to maximize overall utility achievement.

This centralized scheduler requires constructing a global task structure. Figure 9 shows the task structure we generated to match the first experiment setting we used to test different flexibility policies. In the experiment we described in Section 5.1, tasks are generated periodically with random deadlines and random early reward reward rates within specific ranges. Since this centralized scheduler is performing an exhaustive search and cannot handle a task structure with hundreds of tasks, we take one period of 20 time clicks as a sample scenario and model all tasks generated during this time period, which include one *Purchase PC* task and two *Purchase Memory* tasks. Each task is generated with a random deadline and a random early reward reward rate drawn from the same range as the experiment setting described in Section 5.1 . This experiment is repeated for 20 times using the same task structure but each task has different parameter values. The result for centralized approach shown in Table 4 is an average of these 20 experiments. The task structure depicted in Figure 9 is then sent to this centralized scheduler and a multi-agent schedule is generated for each agent in the system. The utility is calculated based on all tasks that are accomplished within the deadline constraints according to such schedule. Additional early reward is also calculated based on the actual finish time, the deadline, and the early reward rate of each task. The sum of the normal reward and the early reward is the actual utility achieved by the system in such time period. This utility then is multiplied by the repetition times (40) of such period (20 time clicks) during the whole distributed experiment setting (800 time clicks), and the result is considered as the total utility the system can achieve when

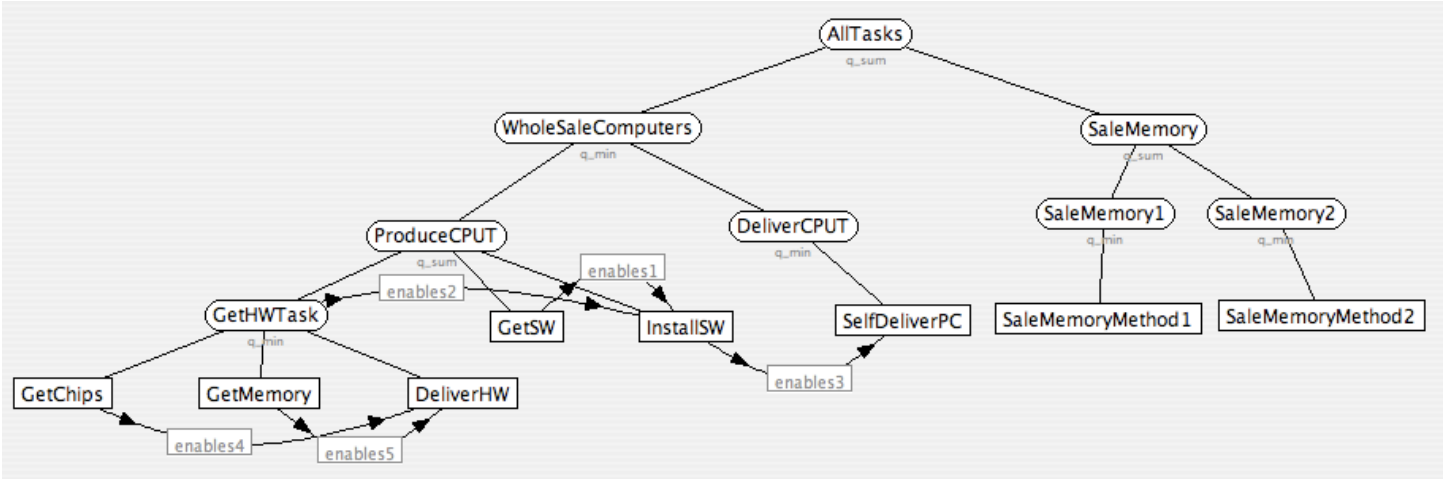

Figure 9: Global Task Structure for the System

Table 4: Comparison With Centralized Approach

Exp. Set #1	Centralized	local-info-flexibility	%	meta-info-flexibility	%		
	1106	338	30.6 %	823	74.4 %		
Exp. Set #2	Centralized	same-deadline	%	meta-info-deadline	%	evenly-divided-deadline	%
	1308	884	67.6%	1192	91.1 %	289	22.1 %

using such a centralized approach. This number is use as a baseline to compare with the distributed approach. A similar process has been performed for the experimental setting described in Section 5.2.

In this periodical modeling approach, it is assumed that there is no time conflict between tasks from different time periods, which is not always true for the continually distributed setting, where the previously committed task may have conflict with a task that arrives later. On the other hand, this periodical modeling approach does not provide the opportunity for agents to interleave the execution of tasks that belongs to different time period, which sometimes happen in a continuous setting. The effects of these two issues actually cancel each other, and further both of them happen very rarely in the continuous experiments. Therefore, the overall effect of the periodical approach is very close to the continuous setting.

However, in this centralized approach, de-commitment penalty is not considered. It is assumed that based on the centralized schedule generated in the beginning, the system can reject all tasks that cannot be handled in time, and hence not paying any de-commitment penalty. Given this assumption, we believe that the centralized approach provides an upper-bound for the performance the system can achieve, which we referred as *optimal performance* in later discussion.

Table 4 describes the comparison of the system performance of the distributed approach with different negotiation meta-strategies to the optimal performance using a centralized approach. In experiment set #1, we compare the system performance achieved when using *local-info-flexibility* policy (338) and *meta-info-flexibility* policy (823) with the system performance achieved by centralized approach (1106). It shows that when agents use the meta-level information to decide how to allocate flexibility (*meta-info-flexibility*), the system achieves 74.4% of the performance achieved by the centralized approach. In experiment set #2, we compare the system performance achieved using different deadline policies with the system performance achieved by centralized approach (1308). When the agents uses the meta-level information both to manage both negotiation deadline and also to allocate flexibilities (*meta-info-deadline*) (1192), the system achieves 91.1% of the performance achieved by the centralized approach.

7 Discussion about Generality

In Section 5 we have shown that the pre-negotiation phase that transfers meta-level information among the agents significantly improves the system's performance under the semi-cooperative system setup described in Section 2. Now we would like to understand how general this conclusion is. Is it still valid for a system where there is no de-commitment penalty or early reward? Can this approach be applied to a completely cooperative system, such as the Coordination Decision Support Assistants (Coordi-

Table 5: Detailed Comparison of Different Negotiation Policies

Policy	tasks received	tasks accepted	tasks cancelled	total penalty paid	total early reward	total utility	w/o penalty	w/o early reward	w/o both
local-info-flex	123	109	90	144	189	338	482	149	293
meta-info-flex	123	103	16	24	370	823	847	453	477
same-deadline	220	148	40	92	319	884	976	565	657
evenly-divided-deadline	220	154	100	253	157	289	542	163	415
meta-info-deadline	220	141	13	15	436	1192	1206	755	770

nator) problem - a DARPA project aiming to create distributed intelligent software systems that will help fielded units adapt their mission plans as the situation around them changes and impacts their plans [16]? In such a completely cooperative system, the performance of each individual agent is not considered, instead, the overall performance of the system is the only key issue that matters. To fully understand these questions, we plot a detailed analysis of the the experimental data in Table 5.

Table 5 shows the system's overall performance under different negotiation policies, including the total number of outside tasks coming in the system, the number of outside tasks accepted by the system and the number of tasks cancelled by the system. These tasks are originally accepted and then cancelled due to the failure to establish all the necessary commitments, which is caused by negotiation failures inside the system. The *system* refers to all the agents located inside the box in Figure 1, the outside tasks are generated by the Customer agent. The utility in Table 5 is the sum of the utilities of individual agents in the system, given the reward of the outside tasks is distributed among these agents. The penalty measures the de-commitment penalty paid towards the outside Customer agent, the inside de-commitment penalty is not counted because it does not affect the system's overall utility. Similarly the early reward measures the early reward the system earns from outside.

Table 5 shows that by using metal-level information to choose local negotiation strategies (**meta-info-flex** and **meta-info-deadline** policies), they system successfully accomplishes more tasks because fewer tasks are cancelled due to negotiation failures. In addition, more tasks are finished earlier, which results in more early reward. The general conclusion is that the better negotiation strategies based on meta-level information increases the system's throughput, in terms of more tasks were finished and finished earlier. The absolute number of the utility, penalty and early reward cannot be generalized, because they depend on the setting of the task reward, penalty rate and early reward rate. However, the data shows that the system's performance is improved even if neither de-commitment penalty nor early reward is considered in the analysis.

We conclude that the more accurate model of negotiation chain is important for the agent to find a better local optimal solution toward the multi-linked negotiation problem in a setting with negotiation chains. A set of better local optimal solutions improves the system's overall performance by allowing more tasks to be accomplished and finished earlier, which indicates that the local scheduling and planning processes are producing solutions that fit better with the global context. This conclusion holds also for complete cooperative systems, where there is no notion for individual reward and de-commitment penalty inside the system. However, even in such system, we feel that some artificial individual reward and de-commitment penalty can be useful in order to communicate how important a task is globally, so that each agent can exploit this information in its local considerations [27].

Another aspect of generality is related to the uncertainty in task execution time. In the experiments described in Section 5, it is assumed that there is no uncertainty in task execution time. In fact, uncertainty can be accommodated in this framework by one of the following two approaches. The first approach is to model uncertainty in the success probability function, a parameter related to uncertainty can be introduced in Formula 6 to ensure that the task with higher uncertainty needs more flexibility in order to succeed. Another approach is to introduce a re-negotiation mechanism which allows the agents to adjust the original commitment within a pre-specified range when task takes longer than expected to finish. Such re-negotiation mechanism has been described in [26], which fits into the overall framework we described here.

8 Related work

Most prior work on negotiation such as [11] studied decision-making process in bilateral negotiation. Fatima, Wooldridge and Jennings [7] studied the multiple issues in negotiation in terms of the agenda and negotiation procedure. However, their work involves only a single agent's perspective without any understanding that the agent may be part of a negotiation chain. Mailler and Lesser [12] have presented an approach to distributed resource allocation problems 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 [20], which is different from our work since the system is a completely cooperative system and individual utility of single agent is not of concern.

Aknine [2] presented a protocol for overlapping negotiation, using a hierarchical contract net model where the negotiations are conducted at different levels. This approach is quite different from ours, where each agent makes it own decisions regarding how to manage multiple related negotiations using meta-level information transferred among agents.

Munroe and Luck [13] has proposed to select dynamically the negotiation opponents based on the consideration of balancing conflict and cost. Selection of negotiation opponents also belongs to issues concerned in the construction of the macro negotiation strategy, the framework proposed in this work can accommodate this decision problem as shown in Figure 3, though we have not implement it in the example scenario. Urbig and Schroter [22] introduced C-IPS approach for negotiation agents for specifying dynamic interdependencies between issues, partners and steps. However, C-IPS approach is mainly focused on one bilateral negotiation, the interdependencies among multiple negotiations was not addressed.

A combinatorial auction [9, 24, 17] 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 [5] 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.

9 Conclusion and future work

In this paper, we have solved distributed negotiation-chain problems by extending our single-agent multi-linked negotiation model to multiple agents. Instead of solving the negotiation-chain problem using a fully centralized approach, we adopt a distributed approach where each agent acquires an extended local model and uses it in its decision-making process. We have introduced a pre-negotiation phase that allows agents to exchange meta-level information on negotiation issues. Using this information, each agent can build a more accurate model of the multi-linked negotiations in terms of modeling the relationship of flexibility and success probability. This more accurate model helps the agent choose the most appropriate local negotiation solution. The agent can also use this information to allocate appropriate time for each local negotiation, so as to find a good ordering of all related negotiations. It turns out that this simple and infrequent exchange of meta-level information and several simple rules actually have profound effect. The experimental data shows that these mechanisms improve the agent's and the system's overall performance significantly, and enables an achievement of 91% of optimal performance achieved by a centralized approach.

The model we describe here can be extended to include the modeling of other agent's negotiation strategies, which affect the negotiation outcomes significantly. Different success probability function models can be built based upon the other agent's specific strategy used in the negotiation, if such knowledge an estimation is available. The agent can choose an appropriate model depending on the negotiation partner. Such improved model has the potential to further improve the agent's local performance and the system's overall performance.

Future extension of this work includes developing mechanisms to verify how reliable the agents are in exchanging meta-level information. Additionally, we would like to develop a learning mechanism that enables the agent to learn how to use the meta-level information from previous experience. Also we would like to introduce some coordinating mediators (agents who are responsible for part of the negotiation chain) and examine whether it would further facilitate the process.

References

[1] TAC SCM game description. http://www.sics.se/tac/page.php?id=14.

[2] Samir Aknine. A multi-agent model for overlapping negotiations. *Group Decision and Negotiation*, pages 1–44, 2011.

[3] 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*, pages 1005–1011, Utrecht, Netherlands, 2005. ACM.

[4] Mike Bryant, Paul Johnson, Brian M Kent, Michael Nowak, and Steve Rogers. Layered sensing: Its definition, attributes, and guiding principles for afrl strategic technology development. http://www.wpafb.af.mil/shared/media/document/AFD-080820-005.pdf.

[5] Wolfram Conen and Tuomas Sandholm. Preference elicitation in combinatorial auctions. In *EC '01: Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 256–259, New York, NY, USA, 2001. ACM.

[6] Frederick Douglis, Michael Branson, Kirsten W. Hildrum, Bin Rong, and Fan Ye. Multi-site cooperative data stream analysis. *Operating Systems Review*, 40(3):31–37, May 2006.

[7] S. Shaheen Fatima, Michael Wooldridge, and Nicholas 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.

[8] Bryan Horling, Regis Vincent, and Victor Lesser. Multi-agent system simulation framework. In *16th IMACS World Congress 2000 on Scientific Computation, Applied Mathematics and Simulation*. EPFL, August 2000.

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

[10] 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.

[11] Guoming Lai, Cuihong Li, and Katia Sycara. Efficient multi-attribute negotiation with incomplete information. *Group Decision and Negotiation*, 15:511–528, 2006. 10.1007/s10726-006-9041-y.

[12] Roger Mailler and Victor 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*, 36(1):80–91, January 2006.

[13] Steve Munroe and Michael Luck. Balancing conflict and cost in the selection of negotiation opponents. *Rational, Robust, and Secure Negotiation Mechanisms in Multi-Agent Systems*, 0:39–54, 2005.

[14] 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. Conoise: Agent-based formation of virtual organisations. *Int. J. Knowledge Based Systems*, 17(2-4):103–111, 2004.

[15] Eugenio Oliveira and Ana Paula Rocha. Agents advanced features for negotiation in electronic commerce and virtual organisations formation processes. In C. Sierra and F. Dignum, editors, *Agent Mediated Electronic Commerce, The European AgentLink Perspective.*, pages 78–97, London, UK, 2001. Springer-Verlag.

[16] DARPA project. Coordination decision support assistants - coordinators. http://www.darpa.mil/ipto/programs/coor/coor.asp.

[17] T. Sandholm. Expressive commerce and its application to sourcing: how we conducted $35 billion of generalized combinatorial auctions. *AI Magazine*, 28(3):45–58, 2007.

[18] Tuomas Sandholm and Victor 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.

[19] Jiaying Shen, Xiaoqin Zhang, and Victor Lesser. Degree of local cooperation and its implication on global utility. In *Proceedings of Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2004)*, pages 546–553, New York, New York, July 2004. IEEE Computer Society.

[20] Mark Sims, Hala Mostafa, Bryan Horling, Haizheng Zhang, Victor Lesser, and Dan 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.

[21] W. T. L. Teacy, N. R. Jennings J. Patel, S. Chalmers M. Luck, N. Oren, T. J. Norman, A. Preece, P. M. D. Gray, G. Shercliff, P. J. Stockreisser, J. Shao, W. A. Gray, N. J. Fiddian, and S. Thompson. Monitoring, policing and trust for grid-based virtual organisations. In *Proc. 4th UK e-Science Meeting, Nottingham*, pages 891–898, Nottingham UK, September 2005.

[22] Diemo Urbig and Kay Schroter. C-IPS approach to negotiating agents: Specifying dynamic interdependencies between issue, partner, and step. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1284–1285, Washington, DC, USA, 2004. IEEE Computer Society.

[23] Thomas Wagner, John Phelps, Valerie Guralnik, and Ryan VanRiper. Coordinators: Coordination managers for first responders. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1140–1147, Washington, DC, USA, 2004. IEEE Computer Society.

[24] W.E. Walsh, M.P. Wellman, and F. Ygge. Combinatorial auctions for supply chain formation. In *Second ACM Conference on Electronic Commerce*, pages 260–269, New York, NY, USA, 2000. ACM.

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

[26] Xiaoqin Zhang, Victor Lesser, and Tom Wagner. A layered approach to complex negotiations. *Web Intelligence and Agent Systems: An International Journal.*, 2(2):91–104, 2004.

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

[28] Qinhe Zheng and Xiaoqin 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.