Efficient Ordering and Parameterization of Multi-Linked Negotiation *

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

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

Sherief Abdallah
Computer Science Department
University of Massachusetts at Amherst
shario@cs.umass.edu

ABSTRACT

Multi-linked negotiation describes a situation where one agent needs to negotiate with multiple agents about different issues (tasks, conflicts, or resource requirements), and the negotiation over one issue influences the negotiations over other issues. In this paper, we present a formalized model of the multi-linked negotiation problem. Based on this model, two search algorithms are described for finding the best ordering of negotiation issues and their parameters. Experimental work is presented which shows that this management technique we developed for multi-linked negotiation leads to improved performance.

General Terms

Keywords

1. INTRODUCTION

Multi-linked negotiation deals with multiple negotiation issues when these issues are interconnected. In a multi-task, resource sharing environment, an agent may need to deal with multiple related negotiation issues including: tasks contracted to other agents, tasks requested by other agents, external resource requirements for local activities and interrelationships among activities distributed among multiple agents.

*This material is based upon work supported by the National Science Foundation under Grant No.IIS-9812755, DMI-0122173 and the Air Force Research Laboratory/IFTD and the Defense Advanced Research Projects Agency under Contract F30602-99-2-0525. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency, Air Force Research Laboratory/IFTD, National Science Foundation, or the U.S. Government.

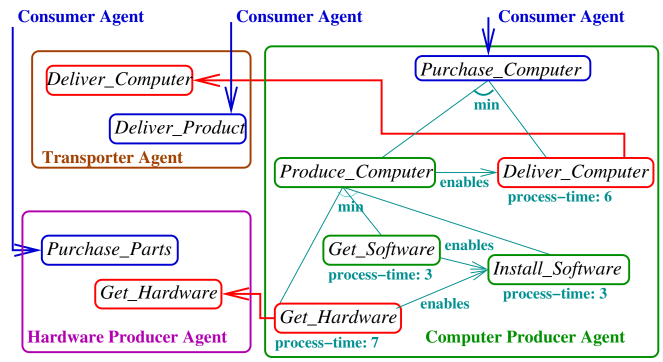

Figure 1: A Supply Chain Scenario

Figure 1 describes a supply chain example[1]. **Consumer Agent** generates orders such as *Purchase_Computer*, *Purchase_Parts* and *Deliver_Product* for different agents including **Computer Producer Agent**, **Hardware Producer Agent** and **Transport Agent**. The agents negotiate about these orders. The *contractor agent* (the agent who performs the task for another agent and gets rewarded for successful completion of the task) and the *contractee agent* (the agent who has a task that needs to be performed by another agent and pays the reward to the other agent) negotiate about the task and a contract is signed (a commitment is built and confirmed) if an agreement is reached during the negotiation. The contract specifies the start time, the finish time, the reward of the task, and other attributes. If the contractor agent can not perform the task as it promised in the contract (i.e. the task could not finish by the promised finish time), it pays a *decommitment penalty* to the contractee agent according to the contract. If the contractee agent needs to cancel the contract after it has been confirmed, it needs to pay a *decommitment penalty* to the contractor agent.

The potential relationships among multiple negotiation issues can be classified as two types. One type of relationship is the *directly-linked* relationship: issue 2 affects issue 1 directly because issue 2 is a necessary resource (or a subtask) of issue 1. The characteristics (such as cost, duration and quality) of issue 2 directly affect the characteristics of issue 1. For example, as pictured in Figure 1, Computer

[1] All task plans shown in this paper use the TÆMS language [1], which is also used in our implementation and experiments.

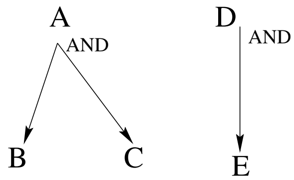

Figure 3: Interrelationships among negotiation issues

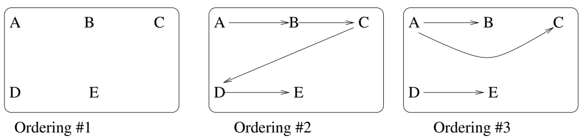

Figure 4: Three possible negotiation orderings

Producer Agent receives task *Purchase_Computer* from Consumer Agent, and decides if it should accept this task and if it does, what the promised finish time of the task should be. The earlier the task is finished, the higher the reward Computer Producer Agent can get. In order to accomplish task *Produce_Computer*, Computer Producer Agent needs to generate an external request for hardware (*Get_Hardware*), and also needs to deliver the computer (*Deliver_Computer*) through a transport agent. The negotiation on the task *Purchase_Computer* is directly linked to the negotiation on the two tasks: *Get_Hardware* and *Deliver_Computer*.

Another type of relationship is the *indirectly-linked* relationship: issue 1 relates to issue 2 because they compete for use of a common resource. For example, as shown in Figure 2, besides the task *Purchase_Computer_A*, Computer Producer Agent has another contract on task *Purchase_Computer_B*. Because of the limited capability of Computer Producer Agent, when task *Purchase_Computer_A* will be performed affects when task *Purchase_Computer_B* can be performed. The negotiation on task *Purchase_Computer_A* and the negotiation on task *Purchase_Computer_B* are *indirectly-linked*. Figure 3 shows the relationships among the five issues for Computer Producer Agent: issue A is directly related to issue B and C, and issue D is directly related to issue E.

How can the agent deal with these multiple related negotiation issues? Two questions need to be answered here. The first question is in what order should the negotiation on each issue be performed. How should the agent orders multiple related negotiations? Should all the negotiations be performed concurrently or in sequence? If in sequence, in what sequence? There are potentially many other choices to order negotiations, such as doing some of them in parallel and some of them in sequence. Figure 4 shows three possible negotiation orderings for the five negotiation issues. Ordering #1 means all five issues are negotiated in parallel; ordering #2 means these five issues are negotiated one by one, first A, then B, then C, then D, and finally E; ordering #3 means that the negotiation on A is performed before the negotiation on B and C, and the negotiation on D is performed before the negotiation on E.

Why is the order of negotiation important? First, because each negotiation issue has a negotiation deadline, set by the contractee agent, if the contractor agent can not reply to a task proposal before the negotiation deadline, the negotiation fails. One reason for missing the negotiation deadline is that the contractor agent is busy on other negotiations before it decides to perform this negotiation. Suppose the negotiation deadline on A is 3 and the negotiations on B and C both take 4 time units, if Computer Producer Agent negotiates on B and C before it replies to Consumer Agent about A, it will lose the contract of A. However, if Computer Producer Agent first replies to Consumer Agent before signing contracts on B and C, there is the risk that the contracts on B and C may not be available or they don't fit with the contract on A, hence Computer Producer Agent needs to pay a decommitment penalty on contract A. How should Computer Producer Agent choose the appropriate negotiation ordering? Secondly, even if the negotiation is completed before its deadline, when the negotiation is started affects the outcome of the negotiation. For example, when there are several potential contractor agents, the earlier a response to negotiation is received, the more likely the offer is accepted. Likewise, the earlier the contractee agent initiates the negotiation, the more likely the contractor agent is to accept the proposal. Thirdly, the earlier a negotiation is started, the larger the space for the agent to find a feasible solution is. For instance, given that the deadline for task *Get_Hardware_A* is 30, if the negotiation on this task finishes at time 10, there are 20 time units left for Hardware Producer Agent to arrange this task; if the negotiation finishes at time 20, Hardware Producer Agent only has 10 time units to execute this task. So the order of negotiation directly affects the outcome of the negotiation.

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

A good negotiation strategy for a multi-linked negotiation problem should provide the agent with an appropriate negotiation order of all issues and a feature assignment (assigns a value to every attribute that needs to be decided in

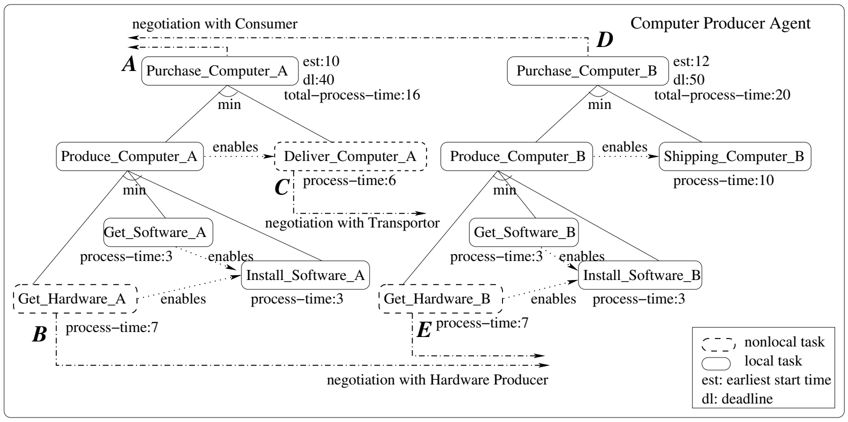

Figure 2: Computer Producer Agent's Tasks

this problem) for those attributes under negotiation[2]. Using such a negotiation strategy, the agent can avoid the conflicts among negotiation issues and optimize its utility achievement. As the experimental results that will be described in Section 5 shown, there can be significantly better performance when optimal sequence among negotiation issues and feature assignment are chosen for the multi-linked negotiation problem as the example shown in Figure 2. Previously, Zhang and Lesser[4] have presented a partial order schedule as a basic reasoning tool for the agent to deal with multi-linked negotiation, which enables the agent to reason about the time-related constraints on each negotiation issue and manage the flexibility. However, this approach does not provide a general solution for how an agent should perform multi-linked negotiations.

To our knowledge, there is no other work that has addressed the *directly-linked* relationship in the negotiation process. There is some work that takes into account the *indirectly-linked* relationship among multiple negotiation issues such as the distributed meeting scheduling [?] problem and the distributed resource allocation problem [?]. However, those problems are different from our problem in the following ways: the negotiation is cooperative by nature and the agent can altruistically withdraw its request to help others succeed; the tasks are simple, no need for subcontracting; no time pressure on negotiation and no penalty for decommitment. The negotiation problem presented in this paper is much more complicate.

In this paper, we introduce a decision-making process that enables an agent to manage the multi-linked negotiation issues and choose the appropriate negotiation strategy based on the knowledge about each negotiation issue and the interrelationships among them. The remainder of this paper is structured in the following manner. Section 2 presents a formalized model of the multi-linked negotiation problem. Section 3 introduces two search algorithms for finding the appropriate negotiation strategy based on the formalized model. Section 4 details how the algorithms and the decision-making process work for a multi-linked negotiation problem using an example. Section 5 reports the experimental work to evaluate this approach, and Section 6 presents the major conclusions and the areas of further work.

2. MODEL OF THE PROBLEM

In this section, we introduce a formalized model of the multi-linked negotiation problem and its solution using the example in Figure 2.

2.1 Definition of the Problem

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

Definition 2.1. *A* **multi-linked negotiation problem** *is defined as an undirected graph (more specifically, a forest as a set of rooted trees): $\mathcal{M} = (V, E)$, where $V = \{v\}$ is a finite set of negotiation issues, and $E = \{(u, v)\}$ is a set of binary relations on V. $(u, v) \in E$ denotes that the negotiation on u and the negotiation on v are directly-linked. Each negotiation issue $v_i (v_i \in V)$ is associated with a set of attributes $\mathcal{A}_i = \{a_{ij}\}$. Each attribute a_{ij} either already has been determined (already has a value) or needs to be decided (to be assigned to a value). The relationships among the negotiation issues are described by a forest, a set of rooted trees $\{T_i\}$. There is a relation operator associated with every non-leaf issue v on the tree (denoted as $\rho(v)$), which describes the relationship between this issue v and its children. This relation operator has two possible values: AND and OR.*

For example, Figure 3 shows the model of the multi-linked negotiation problem (described in Figure 2) for Computer Producer Agent, the problem includes five issues.

A negotiation issue v may be a task to be allocated or a resource to be acquired through negotiation. From an agent's viewpoint, there are two types of negotiation issues:

1. **Incoming negotiation issue**: A task proposed by another agent. For example, issue A (*Purchase_Computer_A*)

[2] The assigned values provide consistent ranges for those attributes under negotiation. For a given negotiation solution, the feature assignment specifies a feasible range or a limit for a feature, such as the feasible range $[9, 36]$ for task *Get_Hardware_A*, or at most 8 units of currency transferred for task *Get_Hardware_A*. The agent can perform a single-step negotiation using this feature assignment as guide, and find the negotiation result quickly. Alternatively, the agent can perform a multi-step negotiation within the limit provided by the value assignment in the negotiation strategy. This detailed negotiation process on a single issue is not discussed in this paper, however, related research work can be found in [5].

and D (*Purchase_Computer_B*) in Figure 2 are incoming negotiation issues for Computer Producer Agent.

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

DEFINITION 2.2. *A negotiation issue v is* **successful** *(denoted as $S(v)$) if and only if a commitment has been established and confirmed for this issue by those agents which are involved in this negotiation.*

DEFINITION 2.3. *A leaf node v is* **task-level successful** *(denoted as $TS(v)$) if and only if v is successful ($S(v) = true$); A non-leaf node v is* **task-level successful** *(denoted as $TS(v)$) if and only if the following conditions are fulfilled:*

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

As in Figure 3, issue A is task-level successful is and only if the negotiation on A is successful, and the negotiations on B and C are also successful. In this case, Computer Producer Agent can actually perform task *Purchase_Computer_A* successfully.

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

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

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

The attributes of a negotiation issue are domain dependent. They are specified according to the application domain. The above functions ζ_i, η_i and θ_i are also defined according to the application domain. The following attributes need to be considered in this supply chain example:

1. time range (st, dl): the time range associated with an issue contains the start time (st) and the deadline (dl). If the issue is a task in negotiation, this task can only be performed during this range (st, dl) to produce a valid result. The larger the time range is, the higher the probability of success this negotiation issue has, since it is easier for the other agent to arrange and schedule this task. For an incoming issue, this range is already determined by the other agent who proposes this request; for an outgoing issue, the agent needs to decide what time range to propose on this issue. The agent needs to make sure this time range is consistent with all other time constraints of other issues, so it can find a feasible local schedule based on the commitment with this time range.
2. duration (d): if the issue is a task in negotiation, duration d is the process time requested to accomplish this task; if the issue is a resource in negotiation, duration d is the time needed for the usage of the resource.
3. flexibility (f): the flexibility is defined based on the time range and the duration: $f = \frac{dl-st-d}{d}$. The flexibility directly affects the *success probability*. For a negotiation issue with negative flexibility, the *success probability* is 0.
4. finish time (ft): the promised finish time for the task. For an incoming issue, the agent needs to decide the promised finish time (which must be no later than deadline dl the other agent requested) for this proposed task when it decides to accept this task.
5. regular reward (r): when an incoming task v is task-level successful, the agent's local utility increases by the amount of $r(v)$.
6. early reward rate (e): for a task in negotiation, if the contractee agent can finish the task earlier than the deadline request, it gets extra reward $e * (dl - ft)$.

The agent needs to find out how these features affect the task-level successful reward and the success probability. These relationships can be described as functions θ, ζ according to the agent's knowledge of each negotiation issue. In this supply chain example, for an incoming negotiation issue (such as A, D), the attribute needed to be decided is the promised finish time ft; the task-level successful reward depends on the promised finish time ft: $\gamma(v) = r(v)+e(v)*(dl(v)-ft(v))$. For an outgoing negotiation issue (such as B, C, and E), the attributes needed to be decided are the start time (st) and the deadline (dl). the *success probability* depends on the flexibility $f(v)$, actually the time range $(st(v), dl(v))$:
$p_s(v) = p_{bs}(v) * (2/\pi) * (\arctan(f(v) + c))$ [3]. p_{bs} is the **basic success probability** of this issue v when the flexibility $f(v)$ is very large. c is a constant parameter used to adjust the relationship.

Besides those domain dependent attributes, there are some common attributes introduced here:

1. negotiation duration ($\delta(v_i)$): the time needed for the negotiation on v_i to get a result, either success or failure.
2. negotiation start time ($\alpha(v_i)$): the start time of the negotiation on v_i. $\alpha(v_i)$ is an attribute that needs to be decided by the agent.
3. negotiation deadline ($\epsilon(v_i)$): the negotiation on v_j needs to be finished before this deadline $\epsilon(v_i)$. The negotiation is no longer valid after time $\epsilon(v_i)$, which is the same as a failure outcome of this negotiation. Furthermore, even if the agent starts the negotiation before $\epsilon(v_i)$, it is not necessarily true that all times before $\epsilon(v_i)$ are equally good. Usually, an earlier started negotiation has a better chance to succeed for two reasons: the other party considers this issue before other later arriving issues, and this issue has a bigger time range for negotiation. This relationship is described by the function ζ_i that takes $\alpha(v_i)$ as one of its parameters.

2.2 Description of the Solution

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

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

concerns the negotiation ordering and the feature assignment.

DEFINITION 2.4. *A* **negotiation ordering** ϕ *is a directed acyclic graph (DAG),* $\phi = (V, E_\phi)$. *If* $e : (v_i, v_j) \in E_\phi$, *then the negotiation on* v_j *can only start after the negotiation on* v_i *has been completed.* $e : (v_i, v_j)$ *is being referred to as a* **partial order relationship (POR),** e. *A negotiation ordering can be represented as a set of* **POR***s,* $\{e\}$.

DEFINITION 2.5. *A negotiation schedule* $\mathcal{NS}(\phi)$ *contains a set of negotiation issues* $\{v_i\}$. *Each issue* v_i *has its negotiation start time* $\alpha(v_i)_\phi$ *and its negotiation finish time* $\varepsilon(v_i)_\phi$ *that is calculated based on its negotiation duration* $\delta(v_i)$ *and its negotiation start time* $\alpha(v_i)_\phi$.

Using the topological sorting algorithm, a negotiation schedule $\mathcal{NS}(\phi)$ can be generated from a negotiation ordering ϕ assuming all negotiation issues are started at their earliest possible times τ for a set of negotiation issues, the negotiation schedule generated from a negotiation ordering is unique. As shown in Figure 4, suppose the negotiation start time $\tau = 0$, and the negotiation duration on each issue is the same $\delta(v_i) = 5$, then the following negotiation schedule is generated for negotiation ordering #3 in Figure 4 according to the assumption that every negotiation issue starts at it earliest possible time:
$A[0,5]B[5,10]C[5,10]D[0,5]E[5,10]$

DEFINITION 2.6. *A feature assignment* φ *is a mapping function that assigns a value* μ_{ij} *to each attribute* a_{ij} *that needs to be decided in the negotiation. For those attributes that already have been decided, value* μ_{ij} *is the decided value.*

DEFINITION 2.7. *A feature assignment* φ *is* **valid** *if given the assigned values of those attributes, there exists at least one feasible local schedule for all tasks and negotiation issues. It is assumed there is a function that can test if a feature assignment* φ *is valid; this function is domain dependent.*

DEFINITION 2.8. *A negotiation strategy* (ϕ, φ) *is a combination of a negotiation ordering* ϕ *and a valid feature assignment* φ.

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

DEFINITION 2.9. *A negotiation outcome* χ *for a set of negotiation issues* $\{v_j\}, (j = 1, ..., n)$ *is a set of numbers* $\{o_j\}(j = 1, ..., n), o_j \in \{0, 1\}$. $o_j = 1$ *means* v_j *is successful,* $o_j = 0$ *means* v_j *fails. There are a total of* 2^n *different outcomes for n negotiation issues, denoted as* $\chi_1, \chi_2, ...\chi_{2^n}$.

DEFINITION 2.10. *The expected value of a negotiation strategy* (ϕ, φ), *denoted as* $\mathcal{EV}(\phi, \varphi)$, *is defined as:* $\mathcal{EV}(\phi, \varphi) = \sum_{i=1}^{2^n} P(\chi_i, \varphi) * (R(\chi_i, \varphi) + C(\chi_i, \phi, \varphi))$

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

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

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

$R(\chi_i, \varphi) = \sum_j \gamma(v_j) = \sum_j \theta_j(\varphi)$
v_j *is a root of a tree and* v_j *is task-level successful* ($TS(v_j) = true$) *according to the outcome* χ_i.

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

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

3. DESCRIPTION OF THE ALGORITHM

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

3.1 Complete Search

ALGORITHM 3.1. *Find the best negotiation strategy.*
Input: $\mathcal{M} = (V, E)$, *the start time for negotiation* τ, *a set of valid feature assignments* $\omega = \{\varphi_k\}$, $k = 1, ..., m$.
The complete search algorithm evaluates each pair of negotiation ordering and valid feature assignment $\mathcal{EV}(\phi_i, \varphi_k)$, *then return the best one.*

If the set of valid feature assignments is a complete set of all possible valid feature assignments, Algorithm 3.1 is guaranteed to find the best negotiation strategy. However, when the attributes have continuous value ranges, it is impossible to find all possible valid feature assignments. We use a depth-first search (DFS) algorithm that searches over the entire value space for all undecided attributes by predefined search step size and finds a set of valid feature assignments[3].

3.2 Heuristic Search

The exponential complexity of algorithm 3.1 prevents it from being used for real-time applications when the number of negotiation issues and the number of valid feature assignments are large; hence a heuristic search algorithm has been developed. The search for the optimal negotiation strategy includes two parts. One is to find the optimal negotiation schedule; the other one is to find the optimal feature assignment for a given negotiation schedule. The search for the optimal negotiation schedule (See Appendix) is based on the simulated annealing idea. Given a negotiation ordering ϕ, randomly pick a POR e, if $e \in E_\phi$, remove it from E_ϕ; otherwise add it into E_ϕ. A new negotiation ordering ϕ_{new} is now generated. If the negotiation schedule $\mathcal{NS}(\phi_{new})$ is better than $\mathcal{NS}(\phi)$, move to ϕ_{new}; otherwise, move to ϕ_{new} with some probability less than 1. This probability decreases exponentially with the "badness" of this move. Three heuristics have been added to this simulated annealing process:

Table 1: Performance of heuristic search algorithm (N.I.: number of negotiation issues; N.F.: Number of feature assignments; Quality: the quality of the approach found by the heuristic search compared to the best approach found by the complete search (with quality of 1.0); C.S.: the number of search steps of the complete search. H.S.: the number of search steps of the heuristic search; Ratio: the ratio of heuristic search steps to complete search steps. N.S.: Number of data samples)

N.I.	N.F.	Quality	C.S.	H.S.	Ratio	N.S.
3	[0, 50)	0.982	336	520	1.547	89
	[50, 100)	1.000	832	590	0.709	3
5	[0, 50)	1.000	1759	1967	1.119	48
	[50, 100)	0.998	3861	1766	0.457	6
6	[0, 50)	1.000	9353	1869	0.200	43
	[50, 100)	0.998	19502	1734	0.089	111
	[100, 150)	0.998	31086	1674	0.054	123
	[150, 200)	0.996	44058	1674	0.038	108
	[200, 250)	0.995	57253	1692	0.030	88
	[250, 300)	0.994	70292	1670	0.024	57
	[300, 350)	0.997	82736	1638	0.020	46
	[350, 400)	0.995	95213	1644	0.017	28
	[400, 450)	0.994	108185	1662	0.015	25
	[450, 500)	0.998	121479	1667	0.014	17

1. Record the best negotiation schedule so far found. When the search process ends, return the best negotiation schedule ever found rather than the current one.
2. Instead of randomly deciding whether to add a POR or remove a POR, use a parameter (*add_por_probability*) to control the probability of the operation "add" or "remove".
3. Instead of completely randomly choosing a POR to change from current negotiation ordering, evaluate every POR e according to how the value of the negotiation schedule changes by adding this POR e to an empty POR set. The probability of adding POR e to the current POR set or removing POR e from the current POR set depends on this evaluation. A POR e with a higher positive evaluation has a higher probability of being added, and has a lower probability of being removed.

The search for the best feature assignment (See Appendix) is based on a hill climbing approach. Randomly pick another feature assignment φ_k. If it is better than current one, move to φ_k. After considering the characteristics of this problem, the following heuristics have been added to this search process:

1. According to the generation process, the change of those valid feature assignments is continuous. Based on this observation, a number of sample points with equal distance (the distance is adjustable, denoted as *sample_step*) in between can be selected from all the valid feature assignments and evaluated. Hill climbing search then can be performed for each sample point.
2. Given current chosen feature assignment, the possible operations include: moving to left and moving to right. If there is a better selection than current one, move to the better selection; otherwise the search stops and a local maxima is found.
3. Compare all local maxima and return the best one.

Experiments are performed to test how good this heuristic algorithm works. In these experiments, the following values were assigned to the parameters: $add\_por\_probability = 0.55$, $TEMP\_MAX = 5$; $TEMP\_STEP = 0.1$; $sample\_step = 10$, $search\_limit = 10^6$. Table 1 shows the performance of this heuristic search algorithm. The quality of the negotiation strategy found is very close to the best strategy found by the complete search. This heuristic search saves a large amount search effort compared to the complete search when the number of negotiation issues and the number of possible feature increase. The heuristic search spends more effort than the complete search when the search space is very small (with a few issues and a few of feature assignments). This problem can be fixed by dynamically choosing the values of the search parameters according to the size of current search space, instead of using the fixed values as we did in these experiments.

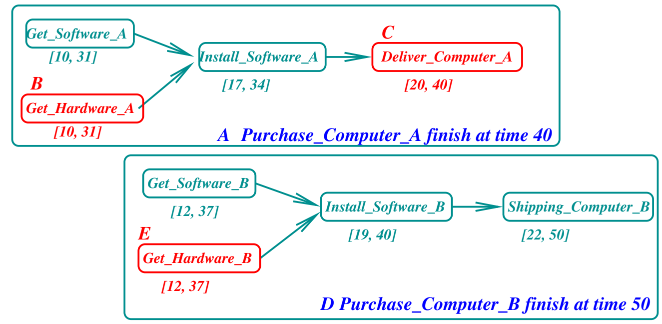

Figure 5: Partial order schedule

4. EXAMPLE

In this section, we demonstrate how the definition and the algorithm work on the supply chain examples in Figure 2. Figure 3 shows the relationships among the five issues for Computer Producer Agent.

For every attribute that needs to be decided: start time (st), deadline (dl) and the promised finish time (ft), the agent can find its maximum possible range using the partial order schedule[4] as shown in Figure 5. The agent searches over the entire possible value space, and use the partial order schedule to test if a feature assignment is valid. A set of valid feature assignments is found and sent to Algorithm 3.1 to find the best negotiation strategy.

To make the output easier to understand, only issues A, B and C are considered in the following example. Table 2 shows the output of Algorithm 3.1 on the example in Figure 3 given following parameters:regular reward $r(A) = 19$;negotiation duration $\delta(A) = 3$, $\delta(B) = 4$, $\delta(C) = 4$; $p_{bs}(B) = 0.95$; $p_s(B) = p_{bs}(B)*(2/\pi)*(\arctan(f(B)+2.5))$; $p_{bs}(C) = 0.99$; $p_s(C) = p_{bs}(C)*(2/\pi)*(\arctan(f(C)+5))$. The different constant parameters for $p_s(B)$ and $p_s(C)$ specify that issue C has a higher *success probability* than issue B given the same flexibility. The following parameters are randomly generated: the *success probability* of A, $p_s(A)$; negotiation deadline ϵ ($\epsilon(A) = \epsilon(B) = \epsilon(C)$); early reward rate $e(A)$; decommitment penalty $\beta(A)$, $\beta(B)$, and $\beta(C)$.

In both case 1 and case 2, the negotiation deadline $\epsilon = 6$ is used, so the valid negotiation ordering has the three negotiation issues performed in parallel. In case 2, issue A has a higher earlier reward rate $e(A)$, and all issues have lower decommitment penalties β than in case 1, so the negotiation

Table 2: Examples of negotiation strategies

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

strategy in case 2 arranges task A to finish 21 time units earlier than the requested deadline, and earns an extra reward of 3.964. In exchange, issues B and C have smaller flexibilities $f(B)$ and $f(C)$, hence lower success probability $p_s(B)$ and $p_s(C)$. In case 3 and case 4, the negotiation deadline $\epsilon = 9$. In case 3, issue A has a much lower success probability $p_s(A)$ than in case 4, so the negotiation on A is scheduled before the negotiation on B and C. In case 5 and case 6, the negotiation deadline $\epsilon = 11$ and the negotiation issues on A, B and C are sequenced according to the success probabilities; the issues with lower *success probability* start earlier. In case 6, issue A has a higher earlier reward rate $e(A)$, and all issues have lower decommitment penalties β than case 5, so the negotiation strategy in case 6 arranges task A to finish 9 time units earlier than the requested deadline; this earns an extra of reward 1.278. In exchange, issues B and C have smaller flexibilities $f(B)$ and $f(C)$ and hence lower success probabilities $p_s(B)$ and $p_s(C)$. It is also important to notice that in all cases, issue B gets larger flexibility than issue C, but has a similar *success probability* to that of issue B. This occurs because it is much easier for issue C to achieve a successful negotiation according to the function that defines the relationship between the *success probability* and the flexibility.

5. EXPERIMENTAL WORK

We have implemented the search and evaluation algorithms so as to enable the reasoning in the multi-linked negotiation process as indicated in the examples described in Section 4. New tasks were randomly generated with decommitment penalty $\beta \in [0, 25]$, early finish reward rate $e \in [0, 0.2]$, and deadline $dl \in [60, 70]$, and arrived at the contractee agents periodically. We use the same task structures as described in Figure 2, tasks vary with randomly generated parameters. In this experiment, Computer Producer Agent needs to deal with the multi-linked negotiation issues related to the incoming task *Purchase_Computer* and the outgoing task *Get_Hardware* and *Deliver_Computer*. The following three different negotiation strategies were tested:

Table 3: Comparison of performance using different negotiation strategies

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

1. Sequenced Negotiation. The agent deals with the negotiation issues one by one, first the outgoing negotiation issues, then the incoming negotiation issues. The finish time promised is the same as the deadline requested from the other agent, and the outgoing issues get the largest possible flexibilities.
2. Parallel Negotiation. The agent deals with the negotiation issues in parallel. It arranges reasonable flexibility (1.5, in this experiment) for each outgoing task, and based on this arrangement, the finish time of the incoming task is decided and promised to the contractee agent.
3. Decision-Based Negotiation. The agent deals with the negotiation as the best negotiation strategy generated by Algorithm 3.1.

The entire experiment contains 40 group experiments. Each group experiment has the system running for 1000 time clicks for three times and each time Computer Producer Agent uses one of the three different approaches. During 1000 time clicks, there are 60 new tasks received by Computer Producer Agent. Table 3 shows the comparison of Computer Producer Agent's performance using different strategies. When the agent uses the sequenced negotiation strategy, more tasks are canceled because of the missed negotiation deadlines. When the agent uses the parallel negotiation strategy, the agent pays a higher decommitment penalty because the failure of the sub-contracted task prevents the incoming task to be task-level successful. The decision-based approach is obviously better than the other two approaches. It chooses to have the negotiation strategy dynamically according to the negotiation deadline and other attributes. Under this experimental setup, it chooses the case where all negotiations are performed in parallel about 13% of the time; it chooses the case where all negotiations performed sequentially about 38% of the time, and the other times it chooses the case where some negotiations are performed in parallel. This strategy enables the agent to get more early reward and pays fewer decommitment penalties.

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

6. SUMMARY

We presented a formalized model of the multi-linked negotiation problem enables the agent to represent and reason about the relationships among different negotiation issues explicitly. Using this model, the best negotiation strategy can be found by a complete search algorithm. A heuristic search algorithm finds the nearly-optimal approach in affordable time. Experimental work shows that this management technique for multi-linked negotiation leads to improved performance. In this work, we model the *success probability* as a function that depends on a set of features, but we have not work out how to construct such a function. In the future, we'd like to use meta-level information and learning technologies for an agent to construct and adjust the structure of this function. Also, in this work, the result of the negotiation is limited to two outcomes: "success" or "fail". Actually, when negotiation is successful, there are potentially many different outcomes depending on the parameters in the commitment. The negotiation process can be modeled as a Markov decision process, and the negotiation strategy can be generated as a policy This is another direction of our future work.

7. REFERENCES

[1] K. S. Decker and V. R. Lesser. Quantitative modeling of complex environments. *International Journal of Intelligent Systems in Accounting, Finance, and Management*, 2(4):215–234, Dec. 1993. Special issue on "Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behavior".

[2] T. Sandholm and V. Lesser. Advantages of a leveled commitment contracting protocol. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 126–133, 1996.

[3] X. Zhang. *Sophisticated Negotiation In Multi-Agent Systems.* PhD thesis, University of Massachusetts Amherst, 2002.

[4] X. Zhang and V. Lesser. Multi-linked negotiation in multi-agent system. In *Proceedings of the First International Joint Conference on Autonomous Agents And MultiAgent Systems (AAMAS 2002).*

[5] X. Zhang, V. Lesser, and R. Podorozhny. New results on cooperative, multistep negotiation over a multi-dimensional utility function. In *AAAI Fall 2001 Symposium on "Negotiation Methods for Autonomous,".*

APPENDIX

ALGORITHM .1. *Find the best negotiation strategy.*
Input: $\mathcal{M} = (V, E)$, *the start time for negotiation* (τ), *a set of valid feature assignments* $\omega = \{\varphi_k\}, k = 1, ..., m$, *the probability to add a por:add_por_probability. TEMP_MAX, TEMP_STEP: search parameters.*
Output:the best negotiation strategy.
begin
  Generate all possible $PORs = \{(v_i, v_j)|v_i, v_j \in V\}$
  total_value = 0;
  total_inverse_value = 0;
  base_value = evaluate_schedule($\mathcal{NS}(V, \emptyset)$*,* ω*);*
  for each por $\in$ *PORs*
    $\phi(por) = (V, por)$
    por.value = evaluate_schedule($\mathcal{NS}(\phi(por))$*,* ω*).value*
    - base_value;
    por.inverse_value = 1.0 / por.value;
    total_value = total_value + por.value;
    total_inverse_value = total_inverse_value + por.inverse_value;
  for each por $\in$ *PORs*
    por.in_probability = por.value/total_value;
    por.out_probability = por.value/total_inverse_value;
  for($t = TEMP\_MAX; t >= 0; t- = TEMP\_STEP$*)*
  generate a random number r between $[0, 1]$*;*
    if(r < add_por_probability)
      choose a por e from PORs/current_ordering
      according to in_probability
      new_ordering = current_ordering $\cap e$
    else
      choose a por e from current_ordering according to out_probability
      new_ordering = current_ordering - e
    evaluation_result = evaluate_schedule($\mathcal{NS}(\phi(new\_ordering))$*,* ω*);*
    change_value = evaluation_result.value - current_value;
    if (change_value $> 0 || random < e^{-change\_value/t}$*)*
      current_value = evaluation_result.value;
      current_assignment = evaluation_result.assignment;
      current_ordering = new_ordering;
    if (change_value > best_value)
      best_value = current_value ;
      best_assignment = current_assignment;
      best_ordering =current_ordering;
  return (best_ordering, best_assignment);
end

ALGORITHM .2. *Evaluate a negotiation schedule with all possible feature assignments and find the best feature assignments and the best value.*
Input: negotiation schedule ϕ*, a set of valid feature assignments* $\omega = \{\varphi_k\}$*,* $k = 1, ..., m$*.*
Output: the best value with the best feature assignment.
begin
  for(i=0; i<=m; i+=sample_step)
    add φ_i *to search_set;*
  for each φ_i *in search_set*
    for(t=0; t<search_limit; t++)
      if($\mathcal{EV}(\phi, \varphi_{i+1}) > \mathcal{EV}(\phi, \varphi_i)$*)*
        i=i+1;
      else if($\mathcal{EV}(\phi, \varphi_{i-1}) > \mathcal{EV}(\phi, \varphi_i)$*)*
        i=i-1;
      else
        break;
    if($\mathcal{EV}(\phi, \varphi_i)$ *> best_value)*
      best_value = $\mathcal{EV}(\phi, \varphi_i)$*;*
      best_assignment = i;
  return(best_value,best_assignment);
end