An Extended Protocol for Multiple-Issue Concurrent Negotiation

Abstract

Negotiation is the technique for reaching mutually beneficial agreement through communication among agents. A concurrent negotiation problem occurs when an agent needs to negotiate with multiple agents to reach agreement. In this paper, we present a protocol to support many-to-many bilateral multiple-issue negotiation in a competitive environment. The protocol is presented in the context of service-oriented negotiation, where one or more self-interested parties will provide services to one or more other parties. By extending existing negotiation protocols, our proposed protocol enables both service requestors and service providers to manage several negotiation processes in parallel. Moreover, this protocol mitigates the situation where most one-to-many negotiations are biased in favor of one participating agent, and reduces the decommitment situation for both participants. We conclude by discussing additional issues related to concurrent multiple-issue negotiation.

Introduction

In supply chains, e-commerce, and Web services, the participants negotiate contracts and enter into binding agreements with each other by agreeing on functional and quality metrics of the services they request and provide. Negotiation is a process by which agents communicate and compromise to reach agreement on matters of mutual interest while maximizing their individual utilities. To meet the requirements of service requestors, multiple issues, including both functional and non-functional, need to be taken into account. Many researchers have investigated multiple-issue negotiation (Fatima, Wooldridge, & Jennings 2004; Jonker & Robu 2004; Dang & Huhns 2005). Fatima et al. presented an optimal agenda and procedure for two-issue negotiation. Dang and Huhns proposed a coalition deal for multiple-issue negotiation to balance between computation cost and negotiation benefit.

Researchers are interested in concurrent negotiation since (1) it is both time efficient and robust when an agent need to negotiate with multiple other agents to make a good deal and (2) it is essential when an agent requests a service involved multiple agents like supply chain problem. Most of recent works focused on one-to-many negotiation. Some (Rahwan, Kowalczyk, & Pham 2002) developed approaches to maximize the service requestor's utility by coordinating multiple concurrent negotiations to exploit the corresponding service providers. Some (Nguyen & Jennings 2004b) proposed the commitment model to enable the agent to break its commitments when an agent receives an offer for a better service in comparison with those for which the agent is committed.

In service-oriented multi-agent environment, it is very likely there are multiple service requestors and providers negotiating simultaneously. We consider a competitive environment and assume the agents are self-interested and know only their own negotiation preferences. Based on two-phase commit protocol (Moss 1985) and the extended CNP protocol (S. Aknine & Shakun 2004), we present a negotiation protocol to support many-to-many multiple-issue negotiation.

This paper advances the state of the art in the following ways. First, most existing protocols of concurrent negotiation do not deal with issues like negotiation consistency and decommitment risk arisen by many-to-many negotiation. Second, most protocols do not deal with the competitive scenario where many issues are involved in and the opponent's preference are unknown. In contract, our proposed protocol (1) enables both service requestors and providers to engagen in several negotiation processes concurrently; (2) avoids the bias towards one participating agent in one-to-many negotiation; (3) improves the negotiation efficiency and robustness by multiple negotiation threads; (4) reduces decommitment situation for both participating agents; (5) provides agents the possibility to adopt hierarchal strategy for concurrent multiple-issue negotiation: the upper coordination level and the lower individual negotiation level.

Presented in the context of service-oriented negotiation, the remainder of the paper is organized as the following: Section 2 discusses the related work. Section 3 describes the negotiation protocol and Section 4 details the algorithms. Section 5 theoretically analyzes the property of the proposed protocol. Section 6 discussed further issues related concurrent multiple-issue negotiation. Section 7 concludes.

Related Work

Negotiation for services involves a sequence of information exchanges between parties to establish a formal agreement among them, whereby one or more parties will provide ser-

vices to one or more other parties. Therefore, Concurrent negotiation is necessary for a service-oriented domain.

The issue of concurrent negotiation is dealt with in (Rahwan, Kowalczyk, & Pham 2002; Nguyen & Jennings 2004a; 2004b). By considering the negotiation as a distributed constraint satisfaction problem, Rahwan (Rahwan, Kowalczyk, & Pham 2002) represented a framework for one-to-many negotiation by conducting a number of coordinated concurrent one-to-one negotiations and discussed the possible negotiation strategies for a coordinator. However, many-to-many negotiation is not equivalent to multiple one-to-many negotiation and issues arising in many-to-many negotiation such as consistency, coordination, and decommitment risk, are too difficult to be handled by the existing protocols.

Nguyen et al. (Nguyen & Jennings 2004a; 2004b) presented a heuristic model for coordinating concurrent negotiation and an integrated commitment model that enable agents reason about when to commit or decommit. In their model, a coordinator on behalf of the buyer manages several negotiation threads one for each individual seller. The buyer first select its strategy for the threads based on its belief, then classify the sellers according to their behaviors during the encounter and consequently adapt the right negotiation strategy based on their classification. Once a thread reaches a deal with a particular seller, the deal is a one-side commitment binding to the corresponding seller and can only be dropped after the buyer finalizes its all negotiation threads. It is obviously biased in favor of the buyer since a commitment should be a bilateral relationship used to bind two participating agents. To mitigate this problem, they allow the seller to de-commit by adopting a commitment model. Both buyer and seller can renege from the previous deal after paying the decommitment penalty. This model still biased in favor of the buyer since any sellers who have already reached a deal have to wait till all negotiation threads end. On the other hand, breaking the commitments is always a hard decision to make because it is usually more issues beyond decommitment penalty need to be considered such as reputation and user feedback etc.

Sandholm and Lesser (Sandholm & Lesser 1995) discussed the automated negotiation among bounded rational self-interested agents in the context of task allocation domain, and presented a protocol to support leveled commitment by introducing the counter-proposal into CNP.

Zhang et al. (Zhang, Lesser, & Abdallah 2004) presented a negotiation mechanism for task allocation in a cooperative system. By two-dimension binary search, agents compromise between their initial proposals and current proposal to generate new proposal alternately and reach an agreement if the marginal gain is more than marginal cost. In (Zhang, Lesser, & Podorozhny 2003), they proposed an approach to deal with multi-linked negotiation in the context of task allocation domain. A partial order scheduler is used to issues in each task and the relationships among them with their flexibilities and dependencies. There is no mutual influence among negotiation threads in their model. Since their protocol does not support concurrent negotiation, it is difficult for a contractee to coordinate among multiple sub-tasks.

In (S. Aknine & Shakun 2004), They proposed an extended version of Contract Net Protocol to support concurrent negotiation processed for the task contractor service provider). It is time efficient and failure tolerant comparing to CNP. However, their protocol did not allow counter-proposing that is very important in negotiation with time constraints especially when multiple issues are involved.

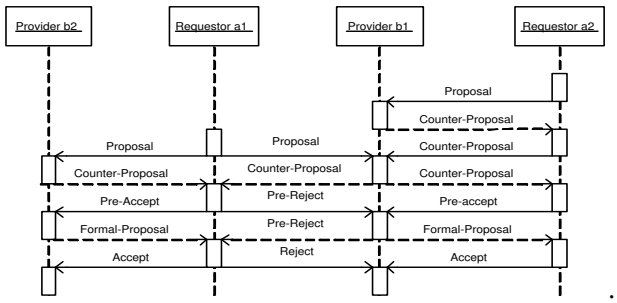

Figure 1: A Concurrent Negotiation Sequence Diagram

Negotiation Protocol

In order to illustrate our protocol, we present a motivating *GetStockQuote* scenario. We assume all agents are self-interested and have their own preferences about the services. Consider a service requestor a_1, might arrange to get a stock quote from a service provider. In this scenario, a_1 locates two service provider b_1 and b_2 that meet its functionality requirements and starts two negotiation processes, one for each service provider, to find the one that provides better service with less cost. Moreover, Consider the situation in which b_1 has already been in a negotiation with another potential service requestor a_2, therefore, b_1 will negotiate with a_1 and a_2 at the same time to find a service requestor who provides the better offer. Most existing protocols can not handle this situation properly. In some protocol, for example, if b_1 is one of the providers that are negotiating with a_1 concurrently, b_1 has to wait till all a_1's negotiation threads end. Even if b_1 reach an agreement with a_1 earlier, it can be accepted or possibly rejected by a_1 only after a_1 finished all his negotiation threads. Therefore, current protocols bound b_1 to a one-side commitment and make b_1 lose time and the potential chance of reaching a contract with other agents, it bias in favor of a_1, but still cause a_1 in the trouble of the likely decommitment to the previously agreed proposals and lead to the loss of utility (decommitment penalty) and reputation that is vital in the future sophisticated negotiation mechanism aiming at long-term cooperation and gains.

By considering the two-phase commit protocol from database system domainand the extended CNP protocol, we introduce two phases of accept and reject into the alternating offers protocol (Osborne & Rubinstein 1994) to support concurrent multiple-issue negotiation. During a negotiation session, a agent can use a number of messages when communicating with its opponent. The negotiation acts are briefly defined in Table 1 and illustrated in a simplified version of our *GetStockQuote* scenario in Figure 2:

In multiple-issue negotiation, different agents have different preferences over the negotiation issues. Their prefer-

Table 1: Negotiation Messages

Proposal	*A requestor initiates the negotiation by proposing an offer for a service.*
Counter-Proposal	*An agent counter-proposes a new proposal in response to the previous proposal*
Formal-Proposal	*An agent formalizes its pre-accepted proposal.*
Pre-Accept	*An agent temporarily accepts a proposal.*
Pre-Reject	*An agent temporarily rejects a proposal.*
Accept	*An agent accepts a proposal.*
Reject	*An agent rejects a proposal.*

ences are usually represented in form of their utility functions with issues as variables. By adapting the alternating offering protocol, an agent makes an offer that gives it the highest utility at the beginning of the negotiation, and then incrementally concedes by offering its opponent a proposal that gives it lower utility as the negotiation progresses.

Let a and b represent the negotiating agents and I a negotiation issue set of n issues, where $I = \{I_1, \ldots, I_n\}$. Given $O_{b\to a,t}$ representing an offer from b to a at time round t, we define agent a's utility as $U_a(O_{b\to a,t})$. The agent b's utility is defined analogously.

Definition 1: In a negotiation where agent a negotiates with a set of agents $B = \{b_1, \ldots, b_n\}$ concurrently, agent b_i's offer $O_{b_i\to a,t}$ is better than agent b_j's offer $O_{b_j\to a,t}$ iff $U_a(O_{b_i\to a,t}) > U_a(O_{b_j\to a,t})$

Definition 2: In a negotiation where agent a negotiates with a set of agents $B = \{b_1, \ldots, b_n\}$ concurrently, agent b_i's offer is acceptable to agent a at time round t if (1) $U_a(O_{b_i\to a,t}) \geq U_a(O_{a\to b_i,t+1})$ and (2) $U_a(O_{b_i\to a,t}) =$ **argmax** $U(O_{b_j\to a,t})$ for $b_j \in B$.

As shown in Figure 2, a requestor agent a_1 locates two provider agent b_1 and b_2, then initiates two negotiation threads with them simultaneously by sending its proposal. After evaluating the received proposal, b_2 sends its counter-proposal to a_1. b_1 is in a negotiation with another requestor agent a_2 when he receives a_1's proposal, b_1 also send its counter-proposal to a_1 since b_1 has not reached any agreement with a_2 yet. After evaluation, a_1 finds that b_2's counter-proposals is acceptable and pre-accepts b_2's counter proposal. a_1 will pre-reject other counter-proposals at the same time. b_1 receives the pre-reject from a_1 and the pre-accept message from a_2, b_1 sends the formal-proposal to a_2 and pre-reject all other requestors. While the pre-accepted b_2 sends a_1 its formal-proposal, other pre-rejected agents send their counter-proposals to a_1. a_1 accepts b_2 and rejects all other providers if b_2's formal-proposal is still acceptable. Analogously, a_2 sends the accept message to b_1 and the reject message to other providers.

Two-phase commitment (*Pre-Accept* and *Accept*) and the corresponding two-phase rejection (*Pre-Reject* and *Reject*) are necessary to deal with the concurrent encounters. With this protocol, the concurrent negotiation process has two phases. In phase one, service agents exchange counter-proposals after service requestors initiate the negotiations. Once receives an acceptable proposal, agent announces that the negotiation phase two comes by sending *pre-accep*t to

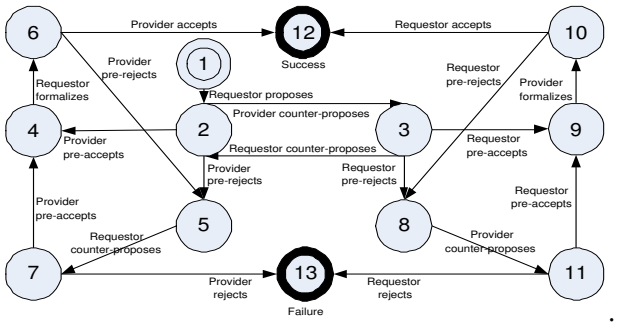

Figure 2: Finite State Machine for Concurrent Negotiation

the agent who send the acceptable proposal and seding *pre-reject* to the rest of negotiating opponents. In phase two, the negotiation enters a process similar to the last-round first-price auction. The pre-accepted agent sends back its formal proposal while other pre-rejected agents send their counter-proposals for their final tries. If the formal proposal is still acceptable, it will be accepted formally and other offers will be rejected to end the negotiation.

Figure 3 depicts a Finite State Machine (FSM) model that describes the concurrent negotiation protocol for services. The left part shows the situation in which the service provider starts the pre-accept or pre-reject phases. The right part shows the situation in which the service requestor starts the pre-accept or pre-reject phases. The service requestor agent starts from state 1 by sending an initial proposal to the service provider agent (state 2). The provider agent evaluates it (state 2), if this proposal is acceptable, it pre-accepts this proposal (state 4); otherwise, it counter-proposes (state 3). Two agents send counter-proposals back and forth before they find an acceptable offer (state 2 and 3). The provider agent may pre-rejects the proposal if it has pre-accepted another agent or has been pre-accepted by another agent (state 5). The pre-accepted requestor sends its formal proposal to the provider (state 6). If this formal proposal is acceptable, it is accepted by the provider (state 12), otherwise, this proposal is pre-rejected (state 5) and the requestor can send its newly built better counter-proposal (state 7), which could be pre-accepted by the provider (state 4) or rejected finally (state 13). The right part of Figure 3 depicts how the service requestor pre-accepts or pre-rejects its peer analogously.

Negotiation Algorithm

In this section, we will describe in detail the negotiation algorithmic of service requestor and provider during a negotiation process. The whole negotiation process has two phases: (1) The proposal exchange phase in which both sides exchange proposals/counter-proposals till one agent sends *Pre-Accept/Pre-Reject* to the other; (2) The proposal formalization phase in which agent sends its formal proposal if it is pre-accepted or the counter-proposal otherwise.

For service requestor, there are three evaluation functions in Figure 4: *EvaluateCounterProposal1*, *EvaluateCounterProposal2*, and *EvaluateFormalProposal*. *EvaluateCounterProposal1* deals with all counter-proposals in phase one.

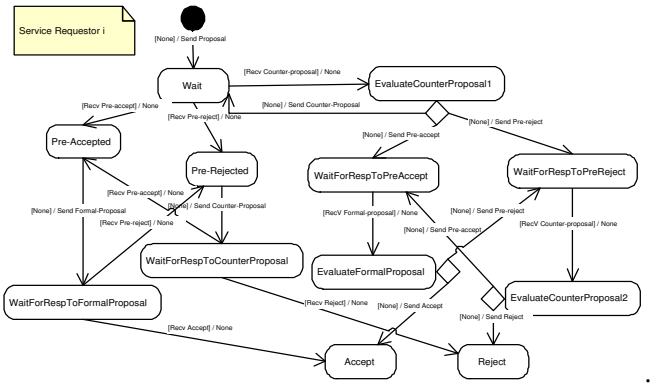

Figure 3: FSM for a Service Requestor

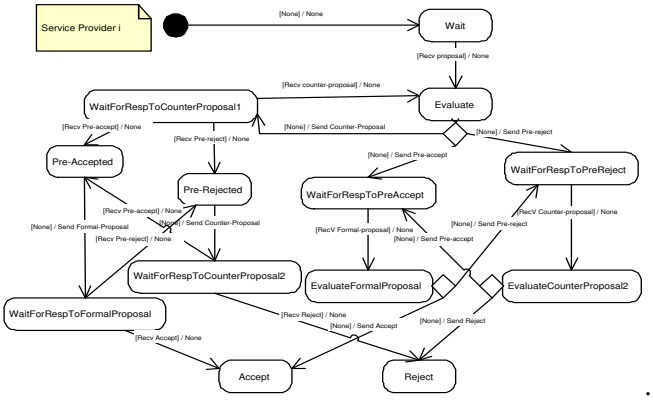

Figure 4: FSM for a Service Provider

It evaluates the counter-proposals from the service providers and send *Pre-Accept*, *Pre-Reject*, or counter-proposal regarding the evaluation result. *EvalutionCounterProposal2* evaluates all counter-proposals in phase two. It sends *Pre-accept* to the sender if its proposal is acceptable, otherwise it sends *Reject*. *EvaluateFormalProposal* evaluates the formal-proposal from the pre-accepted agent, it sends *Accept* to the sender if the formal proposal is acceptable, and it sends *Pre-Reject* otherwise. For service provider, there are also three evaluation functions: *Evaluate*, *EvaluateCounterProposal2*, and *EvaluateFormalProposal*. *Evaluate* deals with all proposals and counter-proposals appear in phase one. It sends the messages of *Pre-Accept*, *Pre-Reject*, or counter-proposals regarding the evaluation result. *EvalutionCounterProposal2* and *EvaluateFormalProposal* are defined as the same as in service requestor's model. Given an agent a and its opponent set $B = \{b_1, \ldots, b_n\}$, We define a set of five flags to indicate the negotiation status between a and b_i. let $f^1_{a \leftrightarrow b_i}$ denotes the negotiation phase one, $f^{2+}_{a \rightarrow b_i}$ denotes that a pre-accepts b_i, $f^{2-}_{a \rightarrow b_i}$ denotes that a pre-rejects b_i, $f^{2+}_{a \leftarrow b_i}$ denotes that a is pre-accepted by b_i, $f^{2-}_{a \leftarrow b_i}$ denotes that a is pre-rejected by b_i.Those flags are exclusive, i.e. only one can be true at a time. Once one flag is ture, others are set to false by default. We assume that the message delivery time is negligible comparing to the time interval of each negotiation round. Messages are sorted and processed in the order of their appearances in Algorithm 1 and the return messages are sent at the end of each round. The detailed algorithm for agent a at time $t < t_d$, where t_d is its negotiation deadline, is defined in Algorithm 1.

Since we assume that the agents are self-interested, it is possible for an agent to propose a very good offer in the phase one in order to scare off its competition and then send a lower formal proposal later. Although it likely is beaten by other agents' counter-proposals, we enforce a negotiation strategy to further avoid this situation. Any formal proposals worse than their pre-accepted proposals will be definitively rejected. The best offer from the received counter-proposals will be pre-accepted as a replacement in this case.

Theoretical Analysis

In this section, we will analyze the properties of the protocol; in particular the termination property and prove that this negotiation process will end after a finite set of steps. Based on the State transition diagram, we'd like to describe the following properties of our concurrent negotiation protocol.

Property 1: Given an agent a and its opponent agent set B, the concurrent negotiation protocol guarantees that agent a can only start the negotiation phase two if the phase one has been completed for all agents in B.

Proof: The negotiation phase two starts when a find an acceptable offer. Since (1) An agent cannot reach a *Pre-Reject* state until one of its peer agents is in a *Pre-Accept* state, and it cannot reach a *Reject* state until one of its peers is in an *Accept* state. It let the rejected/pre-rejected agents know there is at least one competitor that provides better offer than they do. This is enforced by the negotiation algorithm. (2) An agent cannot send its formal-proposal until all its peer agents have received a *Pre-Reject* message, in order to give the pre-rejected peers the chance of sending their counter-proposals to the opponent. There are two paths leading the requestor to the formal-proposal state (1-2-4-6, 1-2-5-7-4-6) in Figure 3. A requestor a needs to be pre-accepted (state 4) to reach the formal-proposal state (state 6). Therefore, all other requestors have received the *Pre-Reject* before a reaches its formal-proposal state since the provider sends a *Pre-Accept* and other requestors *Pre-Reject* simultaneously. It can be proved for the provider in the analogous way.

Property 1 deals with the concurrent encounter in phase two and confirm the consistency during the negotiation. Now we consider the convergence property of the protocol.

Property 2: Given a set of service requestor A and a set of service provider B, the negotiation process engaged by the agents from A and B using the proposed concurrent protocol ends after a finite steps.

Proof: From Figure 3, we can see that loops can occur during the negotiation process. There are three loops: (1) A loop on states 2 and 3, i.e. the service requestor and the provider keep exchanging counter-proposals; (2) A loop on states 5, 7, 4, and 6, i.e. the requestor gets stuck in keep counter-proposing and being pre-accepted, and then being pre-rejected at the formal-proposal state; (3) A loop on state

Notations:
O is a proposal/counter-proposal, $\tilde{O}$ is a formal proposal, PA is *Pre-Accept*, PR is *Pre-Reject*, A is *Accept*, R is *Reject*;
Initialization;
set $f^1_{a\leftrightarrow b_i} = true$ for all $b_i \in B$
if *agent* a *is a requestor* **then**
  sends $O_{a\rightarrow b_i,t_0}$ to all $b_i \in B$
Negotiation;
while $t < t_d$ **do**
  Wait for Message $M_{b_i\rightarrow a}$, $b_i \in B$
  switch *current flag indicator between a and* b_i **do**
    case $f^{2+}_{a\leftarrow b_i} = true$
      if $M_{b_i\rightarrow a} = A_{b_i\rightarrow a}$ **then** negotiation succeeds;
      else if $M_{b_i\rightarrow a} = PR_{b_i\rightarrow a}$ **then** sends $O_{a\rightarrow b_i}$, set $f^{2-}_{a\leftarrow b_i} = true$;
      break;
    case $f^{2-}_{a\leftarrow b_i} = true$
      if $M_{b_i\rightarrow a} = R_{b_i\rightarrow a}$ **then** negotiation fails;
      else if $M_{b_i\rightarrow a} = PA_{b_i\rightarrow a}$ **then** sends $\tilde{O}_{a\rightarrow b_i}$, set $f^{2+}_{a\leftarrow b_i} = true$;
      break;
    case $f^{2+}_{a\rightarrow b_i} = true$
      if $M_{b_i\rightarrow a} = \tilde{O}_{b_i\rightarrow a}$ **then**
        if $\tilde{O}_{b_i\rightarrow a}$ *is acceptable* **then** sends $A_{a\rightarrow b_i}$ to b_i, $R_{a\rightarrow b_j}$ to all $b_j \in B, j \neq i$;
        else sends $PR_{a\rightarrow b_i}$; sends $PA_{a\rightarrow b_k}$ if b_k's offer is acceptable; sends $R_{a\rightarrow b_j}$ to all $b_j \in B, j \neq i, k$; set $f^{2+}_{a\rightarrow b_k} = true$,$f^{2-}_{a\rightarrow b_i} = true$;
      break;
    case $f^{2-}_{a\rightarrow b_i} = true$
      if $M_{b_i\rightarrow a} = O_{b_i\rightarrow a}$ **then**
        if $O_{b_i\rightarrow a}$ *is acceptable* **then** sends $PA_{a\rightarrow b_i}$, sends $PR_{a\rightarrow b_k}$, if b_k's offer is formal-offer; send $R_{a\rightarrow b_j}$ to all $b_j \in B, j \neq i, k$; set $f^{2-}_{a\rightarrow b_k} = true$, $f^{2+}_{a\rightarrow b_i} = true$;
        else sends $R_{a\rightarrow b_i}$
      break;
    case $f^1_{a\leftrightarrow b_i} = true$
      if $M_{b_i\rightarrow a} = O_{b_i\rightarrow a}$ **then**
        if $O_{b_i\rightarrow a}$ *is acceptable* **then** sends $PA_{a\rightarrow b_i}$ to b_i sends $PR_{a\rightarrow b_j}$ to all $b_j \in B, j \neq i$; set $f^{2+}_{a\rightarrow b_i} = true$, set $f^{2-}_{a\rightarrow b_j} = true$ for all $b_j \in B, j \neq i$;
        else sends $O_{a\rightarrow b_i}$
      else if $M_{b_i\rightarrow a} = PR_{b_i\rightarrow a}$ **then**
        sends $O_{a\rightarrow b_i}$, set $f^{2-}_{a\leftarrow b_i} = true$
      else if $M_{b_i\rightarrow a} = PA_{b_i\rightarrow a}$ **then**
        sends $\tilde{O}_{a\rightarrow b_i}$; sends $PR_{a\rightarrow b_j}$ to all $b_j \in B, j \neq i$; set $f^{2+}_{a\leftarrow b_i} = true$; set $f^{2-}_{a\rightarrow b_j} = true$ for all $b_j \in B, j \neq i$
    end
  end
end

Algorithm 1: Negotiation Algorithm

8, 11 , 9 , and 10, the provider gets stuck in the analogous situation as the requestor in (2). To prove that the protocol will ends in a finite step, we must prove that there is no infinite sequence of loops on the above three loops.

(1) In loop 2-3, agents keep exchanging counter-proposal by alternating offering protocol, an agent makes an offer that gives it the highest utility at the beginning of the negotiation, and then incrementally concedes by offering its opponent a proposal that gives it lower utility as the negotiation progresses. Agents have to concede to offer deals that are more likely to be accepted by their opponents if they prefer reaching an agreement to the conflict deal. These principles apply to all concurrent negotiation threads. Many existing mechanism can guarantee agents will keep making progress during the negotiation. With a pre-defined minimum hop, one agent will eventually pre-accept the counter-proposal from its opponent and come out of the loop (1). Also, agent can be kicked out of the loop (1) when its opponent pre-accepts one of its peers from another negotiation threads.

(2) In loop 5-7-4-6, the pre-rejected requestor sends its new counter proposal to the provider and is pre-accepted, however, its formal proposal is then pre-rejected and the requestor has to send a new counter-proposal again. This situation happens when there are existing requestors that keep competing with each other. Let assume a service provider b negotiates concurrently with a set of n service requestors: a pre-accepted requestor a_0 and a set $A' = \{A - a_0\}$ of $n-1$ pre-rejected requestors. After the provider receives the formal proposal from $a0$ and the counter-proposals from A'. Let a_i be the one with the best offer from A'. By comparing the offers from a_0 and a_i, we have:

(a) If $U_b(O_{a_i\rightarrow b}) \leq U_b(O_{a_0\rightarrow b})$, a_0 will be accepted and reach state 12, all requestors from A' will ends with the rejections and reach state 13. The negotiation ends

(b) If $U_b(O_{a_i\rightarrow b}) > U_b(O_{a_0\rightarrow b})$, requestor a_0 will be pre-rejected and reach state 5, requestor a_i will be pre-accepted and enter state 4, all other agents are rejected and are out of the loop. There are only a_i and a_0 left. The negotiation ends if one of them can overbid the other in two consecutive rounds. The infinite loop occurs when a_i and a_0 keep overbidding each other alternately with a tiny amount. It can be prevented by enforcing time constraints to the protocol. Since both sides can get better off if they can reach a contract earlier, the provider should consider the time factor for proposals received in the phase two. For example, before comparing two proposals, a time discount function (e.g. a normalized function whose value decreases exponentially with the time) can be applied to the counter-proposal that needs to evolve two more states to be a formal-proposal. Therefore, the counter proposal need to overbid the formal-proposal more and mre to overrule it along the time and negotiation will end in finite steps.

Concurrent negotiation issues

Negotiation protocol specifies the actions which can be followed by each side to negotiate a contract. This negotiation protocol is more flexible than the extended CNP protocol because agents can exchange counterproposals. The protocol itself does not guarantee the negotiation will end with

a contract. Both participating agents need to adapt certain negotiation strategy to keep the negotiation process going.

With this protocol, concurrent negotiation process can be performed at two different levels: The upper level deals with the coordination among multiple negotiation processes when an agent tries to minimize the possibility of conflicts among different negotiation threads, the lower level deals with the execution of the individual negotiation process.

A negotiation agent consists of two main components: a coordinator and a number of negotiation threads. The coordinator is responsible for coordinating all the threads and choosing an appropriate negotiation strategy for each thread. The negotiation threads deal directly with the various opponents and are responsible for deciding what counter-proposals to send and what proposal to pre-accept. In each round, the threads report their status to the coordinator, the coordinator will use the progress in one negotiation thread to alter the behavior of the agent in another threads.

Since the computation cost becomes crucial when more negotiation threads and issues are involved, some time-efficient strategy like coalition deal can be adopted by the individual negotiation threads to make the negotiation more efficient. Coalition deal can also mitigate the message congestion problem by reducing the computation cost and distributing messages among a number of negotiation threads. We believe that commitment is a binary relationship and address the decommitment problems. Our protocol mitigates the bias and reduces the situation of decommitment for the negotiation participants. From Algorithm 1, we know that our protocol is neutral to both service requestor and provider. Therefore, our protocol can eliminate the situation of decommitment arisen by the one-side commitment.

In negotiation phase two, a service agent host a last-round first-price auction. Since it is the final try for those pre-rejected agents to stay in the negotiation, it makes the truth telling about the reserve offer the dominant strategy for agents whose counter-proposals are close to their reserve offer. At this time, they do not want to lose by providing an offer worse than its reserve offer and they do not want to win the negotiation with the negative gains by providing an offer better than its reserve offer. Therefore, this protocol make the negotiation more efficient. This truth revealing property will be further explored and exploited in future.

Conclusion

This paper investigates the concurrent negotiation in a competitive environment by proposing a negotiation protocol to support many-to-many negotiation among self-interested agents. First, we introduce the many-to-many negotiation problem by a motivating example. Second, we define the protocol communication acts and describe the negotiation protocol. Third, we illustrate the negotiation algorithms, analyze the protocol properties, and in particular prove the termination property of the protocol. Finally, we discuss some issues related to concurrent multiple-issue negotiation.

There are several possible directions for future work. First, we will further investigate the effect of the proposed protocol to agent's negotiation strategy and develop a sophisticated commitment model. Second, we can further extend this protocol to support the negotiation for a composed service with different service agents under the constraints such as QoS and dependency issues among agents.

References

Dang, J., and Huhns, M. N. 2005. Optimal multiple-issue negotiation over qos metrics of web service. *USC CIT Technical Report TR-CIT05-01*.

Fatima, S. S.; Wooldridge, M.; and Jennings, N. 2004. Optimal negotiation of multiple issues in incomplete information settings. In *Proc. Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS'04)*, 1080–1089. New York,USA: ACM.

Jonker, C. M., and Robu, V. 2004. Automated multi-attribute negotiation with efficient use of incomplete preference information. In *Proc. Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS'04)*, 1056–1063. New York,USA: ACM.

Moss, J. E. 1985. *Nested transactions: an approach to reliable distributed computing*. Massachusetts Institute of Technology.

Nguyen, T. D., and Jennings, N. 2004a. Coordinating multiple concurrent negotiations. In *Proc. Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS'04)*, 1064–1071. New York,USA: ACM.

Nguyen, T. D., and Jennings, N. R. 2004b. Reasoning about commitments in multiple concurrent negotiations. In *Proceedings of the 6th International Conference on E-Commerce*.

Osborne, M. J., and Rubinstein, A. 1994. *A Course in Game Theory*. The MIT Press.

Rahwan, I.; Kowalczyk, R.; and Pham, H. H. 2002. Intelligent agents for automated one-to-many e-commerce negotiation. In *CRPITS '02: Proceedings of the twenty-fifth Australasian conference on Computer science*, 197–204. Australian Computer Society, Inc.

S. Aknine, S. P., and Shakun, M. F. 2004. An extended multi-agent negotiation protocol. *International Journal on Autonomous Agents and Multi-agent Systems*.

Sandholm, T., and Lesser, V. 1995. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In Lesser, V., ed., *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*, 328–335. San Francisco, CA, USA: The MIT Press: Cambridge, MA, USA.

Zhang, X.; Lesser, V.; and Abdallah, S. 2004. Efficient Management of Multi-Linked Negotiation Based on a Formalized Model. *Autonomous Agents and Multi-Agent Systems*.

Zhang, X.; Lesser, V.; and Podorozhny, R. 2003. Multi-Dimensional, MultiStep Negotiation for Task Allocation in a Cooperative System. *Autonomous Agents and MultiAgent Systems (conditionally accepted for publication)*.