An Extended Protocol for Multiple-Issue Concurrent Negotiation

Jiangbo Dang and **Michael N. Huhns**
University of South Carolina
Columbia, SC 29208 USA
{dangj,Huhns}@sc.edu

Abstract

Negotiation is a technique for reaching mutually beneficial agreement among agents via communication. 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 can provide services to one or more other parties. By extending existing negotiation protocols, our described protocol enables both service requestors and service providers to manage several negotiation processes in parallel. Moreover, this protocol mitigates the situation where most known one-to-many negotiations are biased in favor of one participating agent, and allow the negotiation participants to make durable commitments to reduce the decommitment situation. 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 that might be functional or nonfunctional need to be taken into account. Many researchers have investigated multiple-issue negotiation. Fatima et al. (Fatima, Wooldridge, & Jennings 2004) presented an optimal agenda and procedure for two-issue negotiation. (Dang, Shrotri, & Huhns 2005) described a coalition deal for multiple-issue negotiation that balances computation cost and negotiation benefit.

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

In a service-oriented multiagent 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 the two-phase commit protocol (Gray 1978) from database transaction theory 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 for concurrent negotiation do not deal with issues such as negotiation consistency and decommitment risk, which arise in many-to-many negotiation. Second, most protocols do not deal with a competitive scenario where many issues are involved and the opponent's preferences are unknown. In contrast, our new protocol (1) enables both service requestors and providers to engage in several negotiation processes concurrently; (2) avoids the bias towards one participating agent in one-to-many negotiation; (3) improves negotiation efficiency and robustness; (4) reduces the number of decommitment situations for both participating agents; and (5) provides agents the possibility to adopt a hierarchal strategy for concurrent multiple-issue negotiation: an upper coordination level and a lower individual negotiation level.

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

Related Work

Negotiation for services involves a sequence of information exchanges among 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 negotiation as a distributed constraint satisfaction problem, the framework of (Rahwan, Kowalczyk, & Pham 2002) supports one-to-many negotiation by coordinating a number of concurrent one-to-one negotiations and allows several 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 existing protocols.

Nguyen et al. (Nguyen & Jennings 2004a; 2004b) present a heuristic model for coordinating concurrent negotiation and an integrated commitment model that enable agents to 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 selects its strategy for the threads based on its beliefs, then classifies the sellers according to their behaviors during the encounter and consequently adapts the right negotiation strategy based on their classification. Once a thread reaches a deal with a particular seller, the deal is a one-sided commitment binding on the corresponding seller and can only be dropped after the buyer finalizes all its 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 decommit by adopting a commitment model, and then both buyer and seller can renege on the previous deal by paying the decommitment penalty. This model is still biased in favor of the buyer, since any sellers who have already reached a deal have to wait until all negotiation threads end. Also, incorporating seller decommitment into their model has no advantage, because in one-to-many negotiation there are no other buyers for a seller, so it has no incentive to decommit and will never do so rationally. On the other hand, breaking commitments is always a hard decision to make, because usually more issues beyond a decommitment penalty need to be considered, such as reputation and user feedback.

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

Zhang et al. (Zhang, Lesser, & Abdallah 2004) present a negotiation mechanism for task allocation in a cooperative system. By two-dimensional binary search, agents compromise between their own proposal and their opponent's current proposal to generate new proposals alternately and reach an agreement if the marginal gain is more than the marginal cost. In (Zhang, Lesser, & Podorozhny 2003), they describe an approach to deal with multilinked negotiation in the context of a task allocation domain. A partial-order scheduler is used to oreder the 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 the protocol does not support concurrent negotiation, it is difficult for a contractee to coordinate among multiple subtasks.

Table 1: Negotiation Messages

Proposal	*A requestor initiates the negotiation by proposing an offer for a service*
Counterproposal	*An agent counterproposes 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*

In (S. Aknine & Shakun 2004), an extended version of the Contract Net Protocol is presented to support concurrent negotiation processes for a task contractor (service provider). It is efficient and failure tolerant compared to CNP. However, the protocol does not allow counterproposals, which is very important in negotiations involving time constraints, especially when multiple issues are involved.

Negotiation Protocol

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 they need. A service requestor a_1 might arrange to get a stock quote from a service provider. In this scenario, a_1 locates two providers b_1 and b_2 that meet its functionality requirements and starts two negotiation processes, one for each provider, to find the one that provides better service with lower cost. For the situation in which b_1 has already been in a negotiation with another potential service requestor a_2, b_1 will negotiate with a_1 and a_2 at the same time to find a requestor that provides a better offer. Most existing protocols cannot handle this situation properly. In some protocols, for example if b_1 is one of the providers that are negotiating with a_1 concurrently, b_1 has to wait until all a_1's negotiation threads end. Even if b_1 reaches an agreement with a_1 earlier, it can be accepted or possibly rejected by a_1 only after a_1 finishes all its negotiation threads. Therefore, current protocols bind b_1 to a one-sided commitment and make b_1 lose time and the potential chance of reaching a contract with other agents. This is biased in favor of a_1, but still causes a_1 the trouble of a likely decommitment to the previously agreed proposals, and leads to the loss of utility (decommitment penalty) and reputation that would harm an agent interested in long-term cooperation and gains.

By considering the two-phase commit protocol and 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, an 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* sce-

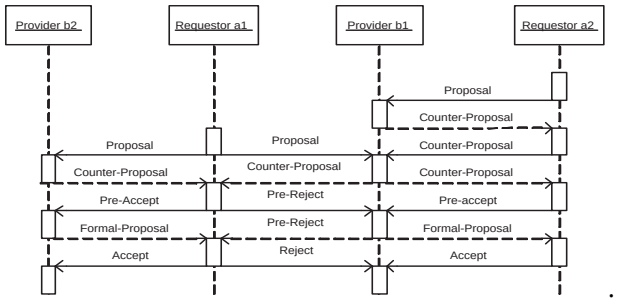

Figure 1: A Concurrent Negotiation Sequence Diagram

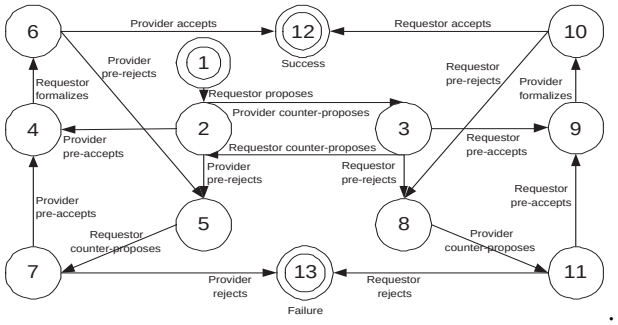

Figure 2: Finite State Machine for Concurrent Negotiation

nario in Figure 1.

In multiple-issue negotiation, different agents have different preferences over the negotiation issues. Their preferences are usually represented in the form of their utility functions with issues as variables. By adopting the alternating offer 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 set of n negotiation 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 $U_a(O_{b\to a,t})$. 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 1, a requestor agent a_1 locates two provider agents b_1 and b_2, then initiates two negotiation threads simultaneously by sending them its proposal. After evaluating the received proposal, b_2 sends its counterproposal to a_1. b_1 is in a negotiation with another requestor agent a_2 when it receives a_1's proposal. b_1 also sends its counterproposal to a_1, since b_1 has not reached any agreement with a_2 yet. After evaluating it, a_1 finds that b_2's counterproposal is acceptable and pre-accepts b_2's counter proposal. a_1 will pre-reject other counterproposals at the same time. b_1 receives the pre-reject from a_1 and the pre-accept message from a_2, so b_1 sends the formal-proposal to a_2 and pre-rejects all other requestors. While the pre-accepted b_2 sends a_1 its formal-proposal, other pre-rejected agents send their counterproposals 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 of (*Pre-Accept* and *Accept*) and the corresponding two-phase rejection (*Pre-Reject* and *Reject*) are necessary to deal with concurrent encounters. With this protocol, the concurrent negotiation process has two phases. In phase one, service agents exchange counterproposals after service requestors initiate the negotiations. Once an agent receives an acceptable proposal, the agent announces that the negotiation phase two begins by sending *pre-accept* to the agent who sent the acceptable proposal and sending *pre-reject* to the rest of the negotiating opponents. In phase two, the negotiation enters a process similar to a last-round first-price auction. The pre-accepted agent sends back its formal proposal while other pre-rejected agents send their counterproposals for their final tries. If the former proposal is still acceptable, it will be accepted formally and other offers will be rejected to end the negotiation.

Figure 2 depicts a Finite State Machine 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) and if this proposal is acceptable, the provider agent pre-accepts it (state 4); otherwise, the agent counterproposes (state 3). Two agents send counterproposals back and forth before they find an acceptable offer (states 2 and 3). A provider agent may pre-reject a 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 an improved counterproposal (state 7), which could be pre-accepted by the provider (state 4) or rejected finally (state 13). The right part of Figure 2 depicts how the service requestor pre-accepts or pre-rejects its peer analogously.

Negotiation Algorithm

In this section, we describe in detail the negotiation algorithms of a 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/counterproposals until one agent sends *Pre-Accept/Pre-Reject* to the other; (2) The proposal formalization phase in which an agent sends its formal proposal if it is pre-accepted or the counterproposal otherwise.

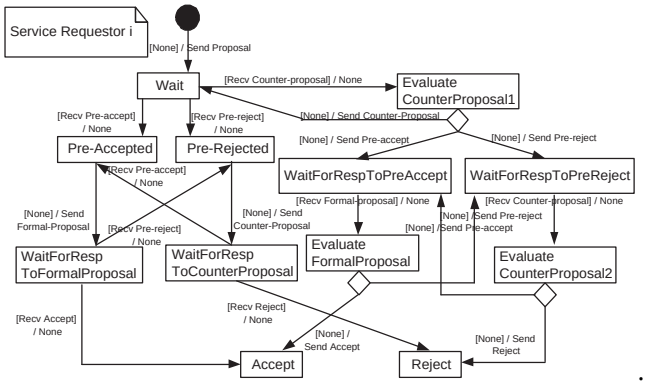

Figure 3: Finite-State Machine for a Service Requestor

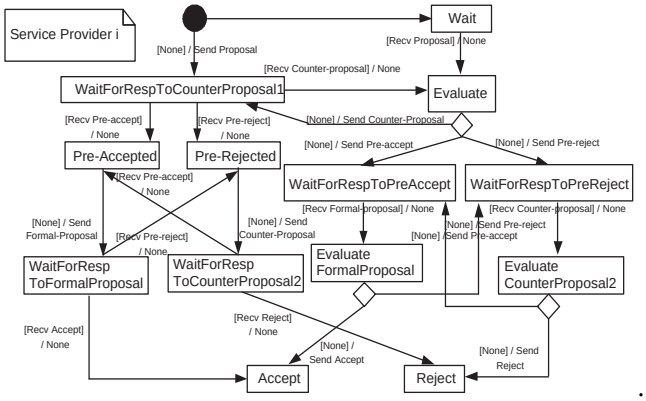

Figure 4: Finite-State Machine for a Service Provider

For a service requestor, there are three evaluation functions in Figure 3: *EvaluateCounterProposal1*, *EvaluateCounterProposal2*, and *EvaluateFormalProposal*. *EvaluateCounterProposal1* deals with all counterproposals in phase one. It evaluates the counterproposals from the service providers and sends *Pre-Accept*, *Pre-Reject*, or a counterproposal regarding the evaluation result. *EvalutionCounterProposal2* evaluates all counterproposals 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 a service provider there are also three evaluation functions: *Evaluate*, *EvaluateCounterProposal2*, and *EvaluateFormalProposal*. *Evaluate* deals with all proposals and counterproposals that appear in phase one. It sends the messages of *Pre-Accept*, *Pre-Reject*, or counterproposals regarding the evaluation result. *EvalutionCounterProposal2* and *EvaluateFormalProposal* are defined the same as in the 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 $f^1_{a \leftrightarrow b_i}$ denote 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, and $f^{2-}_{a \leftarrow b_i}$ denotes that a is pre-rejected by b_i. These flags are exclusive, i.e., only one can be true at a time. Once one flag is true, others are set to false by default. We assume that the message delivery time is negligible compared 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 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' counterproposals, 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 counterproposals will be pre-accepted as a replacement in this case.

Theoretical Analysis

In this section we analyze the properties of the protocol, particularly the termination property, and prove that our negotiation process will end after a finite number of steps. Based on the state-transition diagram, we 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 negotiation phase two if phase one has been completed for all agents in B.

Proof: Negotiation phase two starts when a finds an acceptable offer. (1) Since 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 lets 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 counterproposals 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 2. 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 analogously.

Property 1 deals with a concurrent encounter and confirms consistency during the negotiation. Now we consider the termination property of the protocol.

Property 2: Given a set of service requestors A and a set of service providers B, a negotiation process engaged by the agents from A and B using our concurrent protocol ends after a finite number of steps.

Proof: From Figure 2, we can see that three loops can occur during the negotiation process: (1) A loop on states 2 and 3, i.e., the requestor and the provider keep exchanging coun-

Notations:
O is a proposal/counterproposal, $\tilde{O}$ 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

terproposals; (2) A loop on states 5, 7, 4, and 6, i.e., the requestor gets stuck counterproposing and being pre-accepted, and then being pre-rejected at the formal-proposal state; (3) A loop on state 8, 11 , 9 , and 10, where the provider gets stuck in the analogous situation as the requestor in (2). To prove that the protocol will end in a finite number of steps, we must prove that there is no infinite sequence of loops on the above three loops.

(1) In loop 2-3, agents keep exchanging counterproposals by the alternating offering protocol, in which 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 mechanisms can guarantee agents will keep making progress during the negotiation. With a predefined minimum hop ϵ, one agent will eventually pre-accept the counterproposal from its opponent and exit from the loop (1) or time out. Also, an 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 counterproposal to the provider and is pre-accepted; however, its formal proposal is then pre-rejected and the requestor has to send a new counterproposal again. This situation happens when there are existing requestors that keep competing with each other. Let us 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 a_0 and the counterproposals 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 and all requestors from A' will end with 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, and all other agents are rejected and 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. An infinite loop occurs when a_i and a_0 keep overbidding each other alternately by a tiny amount. It can be prevented by defining a minimum increment ϵ or enforcing time constraints on the protocol. Since both sides would be better off if they can reach a contract earlier, the provider should consider the time factor for proposals received in phase two. For example, before comparing two proposals, a time discount function $\delta(t) < 1$ (e.g., a normalized function whose value decreases exponentially with the time) can be applied to the counterproposal that needs to evolve two more states to be a formal-proposal. Therefore, the counterproposal needs to overbid the formal proposal more and more to overrule it as the protocol proceeds, and negotiation will end in a finite number of steps.

Concurrent Negotiation Issues

Our protocol specifies the actions that 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 adopt a certain negotiation strategy to keep the process going.

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

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 solving the conflicts among them. The negotiation threads deal directly with the various opponents and are responsible for deciding what counterproposals to send and what proposal to pre-accept. In each round, the threads report their status to the coordinator, and the coordinator will update the status of the other threads and use the progress in one thread to alter the behavior of the agent in the other threads.

Since the computation cost becomes crucial when more negotiation threads and issues are involved, some strategy such as a coalition deal can be adopted by the individual negotiation threads to make the negotiation more efficient. A 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 symmetric relationship among the participants in a negotiation. Our protocol allows the negotiation participants to make durable commitments to reduce the likelihood of a decommitment situation. Algorithm 1 shows that our protocol is neutral to both service requestors and service providers. Therefore, our protocol can eliminate the decommitment situations arising in one-sided commitment.

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

Conclusion

This paper investigates 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's communication acts and describe the negotiation protocol. Third, we illustrate the negotiation algorithms, analyze the properties of the protocol, 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 this protocol to an agent's negotiation strategy and develop a sophisticated commitment model. Second, we can extend this protocol to support negotiation for a composed service with different service agents under constraints such as QoS and dependency issues among agents.

References

Dang, J.; Shrotri, D.; and Huhns, M. N. 2005. Distributed coordination of an agent society based on obligations and commitments to negotiated agreements. In Scerri, P., ed., *Challenges in the Coordination of Large-Scale Multiagent Systems*. Springer Verlag.

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.

Gray, J. 1978. Notes on data base operating systems. In Bayer, R.; Graham, R. M.; and Seegmuller, G., eds., *Operating Systems, An Advanced Course*. Springer-Verlag: London, UK. 393–481.

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 *Proc. 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: Proc. of the twenty-fifth Australasian conference on Computer science*, 197–204.

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 *Proc. of the First International Conference on Multi-Agent Systems (ICMAS'95)*, 328–335.

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*.