Multi-Dimensional, MultiStep Negotiation for Task Allocation in a Cooperative System *

Xiaoqin Zhang
Department of Computer and Information Science
University of Massachusetts at Dartmouth
Victor Lesser
Department of Computer Science
University of Massachusetts at Amherst
Rodion Podorozhny
Department of Computer Science
University of Texas at Austin

January 24, 2004

Abstract

We present a multi-dimensional, multi-step negotiation mechanism for task allocation among cooperative agents based on distributed search. This mechanism uses marginal utility gain and marginal utility cost to structure this search process, so as to find a solution that maximizes the agents' combined utility. These two utility values together with temporal constraints summarize the agents' local information and reduce the communication load. This mechanism is anytime in character: by investing more time, the agents increase the likelihood of getting a better solution. We also introduce a multiple attribute utility function into negotiations. This allows agents to negotiate over the multiple attributes of the commitment, which produces more options, making it more likely for agents to find a solution that increases the global utility. A set of protocols are constructed and the experimental result shows a phase transition phenomenon as the complexity of negotiation situation changes. A measure of negotiation complexity is developed that can be used by an agent to choose an appropriate protocol, allowing the agents to explicitly balance the gain from the negotiation and the resource usage of the negotiation.

Keywords: Cooperative Negotiation; Distributed Search; Multi-Agent System;

1 Introduction

Negotiation is a process by which two or more parties make a joint decision. The agents first verbalize demands and then move toward an agreement through a process of concession formation or search for new alternatives [1]. Negotiation research in

* This material is based upon work supported by the National Science Foundation under Grant No. IIS-9812755 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.

multi-agent systems falls into two main categories, competitive negotiation and cooperative negotiation. Competitive negotiation occurs among self-interested agents [2], each trying to maximize its local utility; while in cooperative negotiation, agents try to reach the maximum global utility that takes into account the worth of all their activities. This latter form of negotiation is quite different from competitive negotiation, and can be viewed as a distributed search process. We will focus on this cooperative negotiation which, as of late, has not received very much attention in the related literature [7]. In fact, we feel there is very little work on cooperative negotiation that explicitly tries to maximize a multi-dimensional global utility function. Chia, Neiman and Lesser's work on the Distributed Dynamic Scheduling System (Dis-DSS) [4] has shown that the coordination among agents increase schedule quality; though no explicit negotiation using a cost model was involved in this work. The closest work to our knowledge is that of Moehlman et al. [9]; however their work involves a much simpler and more structured utility function that does not involve quantitative reasoning about the combined utility of the agents. Additionally, their approach is not empirically evaluated in different negotiation situations. Task allocation among cooperative agents was also studied by Shehory and Kraus [11]. However, their work was focused on agent coalition formation, not negotiation and also the tasks were much simpler without alternative approaches.

There are different degrees of cooperation in a multi-agent system. The most extreme is "global cooperation", which occurs when an agent, while making its local decision, always tries to maximize the global utility function that takes into account the activities of all agents in the system. Global cooperation is unachievable in most realistic situations because of the number of agents and bounds on computational power and bandwidth. Thus we focus our research on "local cooperation" [8] which occurs when two or more agents, while negotiating over an issue, try to find a solution that increases the sum of their local utilities, without taking into account the rest of the agents in the system. Furthermore, our agents negotiate over multiple attributes (dimensions) rather than over a single dimension. For example, agent A wants agent B to do task T for it by time 10, and requests the minimum quality of 8 for the task to be achieved. Agent B replies that it can do task T by time 10 but only with the quality of 6; however, if agent A can wait until time 15, it can get a quality of 12. Agent A will select the alternative it believes is better for both agents. The negotiation relates to both the completion time and achieved quality of the task, and thus the scope of the search space for the negotiation is increased, improving the agents' chance of finding a solution that increases the combined utility.

Our approach focuses on a multi-step negotiation process in which agents engage in a series of proposals and counter-offers to decide whether the contractor agent will perform a task for the contractee agent by the specified time with a certain quality. This is a search for those plans and constructed schedules of an agent's local activities that increase or maximize the combined utility of the agents. We will use measures of marginal gain and marginal cost first used in the TRACONET agents [10] to structure the search. In that work, these measures were used for a single phase evaluation rather than as a basis for a cooperative/distributed search among the agents to find the best combined local schedule.

The cooperative negotiation process can potentially have many outcomes, depending upon the amount of effort that the agents want to expend on the negotiation. One possibility is that they will find a solution that leads to the maximum combined utility. Another possibility is that they will find a solution that increases the combined utility from their current state. While a third possibility is that they may find that either there is no solution that increases the combined utility or that they can not find one given a limited search[1].

[1] Another possibility, which we will not consider in this work, is that the agents will recognize either at the start of negotiation or at some intermediate point that either it is highly unlikely that a solution that increases the global utility would be found or the effort to find such a solution is not worthwhile in the current context. In this case, each agent could enter a meta-level phase of the negotiation process where it could

After the negotiation starts, an agent needs to decide when to stop the process because negotiation costs accrue with time. It may stop after it gets the first acceptable solution that increases the joint utility or it may decide to continue looking for a better one. The agent needs to establish a balance between the negotiation cost and the negotiation benefit. There are many different possible variations of cooperative negotiation protocol, depending on the stopping criteria. Therefore, as part of this work we will examine these questions experimentally to produce insights about how the characteristics of the current situation affect the variant of the protocol chosen. In the experiments, we will use the TÆMS language [5] to represent the agent's local tasks and activities (See Figure 2)[2], and the DTC (Design-To-Criteria)[12][3] scheduler generates a local schedule for the agent, it attempts to maximize the agent's local utility based on a specified multi-dimensional utility function.

In the remainder of the paper, we present our work on cooperative negotiation in the task allocation domain. We first describe the negotiation framework, followed by the cooperative negotiation protocol for task allocation. We next discuss the experimental results obtained by using these protocols. Finally, we summarize our work and discuss future work.

2 Task Allocation Negotiation Mechanism

In a multi-agent system, an agent may need to contract out one of its tasks to another agent if it does not have the capabilities to perform this task locally, or if it is overloaded, or if the other agent can do the job better. This task can potentially be part of a larger activity that the agent performs in order to achieve some desired goal. In order to accomplish this task, the agent needs to negotiate with another agent about the appropriate time and approach to execute this task, so that the combined utility (the sum of both agent's local utilities) can be increased. By "approach", we mean a specific way for an agent to perform the task which might differ in the resources (i.e. the computation time and cost) used and the quality of the solution obtained from other alternative ways. We assume that the agent will communicate with the negotiation subsystem about which task it definitely can't do locally and those tasks that it thinks may be advantageous to be performed by another agent. As part of the negotiation process, the relative merits of doing the task locally, not doing this task or contracting the task to another agent, will be taken into account.

2.1 Definitions

- Contractee Agent (contractee): the agent which has a task (non-local task NL) that needs to be assigned to another agent. The contractee gains quality from this task when it is completed (TCE is the contractee's local task structure).
- Contractor Agent (contractor): the agent which performs this task for the contractee. It devotes processing time and other resources to this task without directly gaining quality (TCR is the contractor's local task structure).

either abandon the negotiation or change the context of the negotiation by altering the set of objective criteria issues over which agents negotiate. Thus, before the negotiation, an agent could evaluate the current situation to decide if it should start the negotiation based on whether it has a good chance of increasing the global utility.

[2]The TÆMS task modeling language is a domain-independent framework used to model the agent's potential activities. It is a hierarchical task representation language that features the ability to express alternative ways of performing tasks, statistical characterization of methods via discrete probability distributions in three dimensions (quality, cost and duration), and the explicit representation of interactions between tasks.

[3]It is a domain-independent scheduler that aims to find a feasible schedule that matches the agent's local criteria request. The first input for the DTC scheduler is the TÆMS task structure that describes the agent's local activities and the objective criteria used to evaluate alternative schedules. The second input is a set of existing and proposed commitments, C, that indicates that this agent will produce specific results of certain qualities by certain times. The third input is a set of non-local commitments, NLC, that are commitments made to this agent by other agents. The scheduler uses this information to find the best schedule given the objective criteria, that exploits the given non-local commitments, honors the existing commitments and satisfies the proposed commitments as best as possible.

- Marginal Utility Gain [NL, C] (MUG): the local utility increment for the contractee by having task NL performed with duration and quality specified by commitment C, which is calculated by the contractee agent.
- Marginal Utility Cost [NL, C] (MUC): the local utility decrement for the contractor by performing task NL with duration and quality specified as in commitment[4],which is calculated by the contractor agent.

2.2 Mechanism description

Figure 1 depicts a Finite State Machine (FSM) model that describes the agents' protocol for the task allocation mechanism. The upper part shows the contractee's FSM; the lower part shows the contractor's FSM. The contractee agent starts the negotiation by building a proposal (Action A:*buildProposal*) and sending this proposal (Action B:*sndMsgProposal*) to the contractor agent. After receiving this proposal (*rcvMsgProposal*), the contractor agent evaluates it (Action J:*evalProposal*): if the marginal utility cost is less than the marginal utility gain, it accepts this proposal (Action M:*sndMsgAccept*); otherwise, this proposal is rejected, the contractor agent builds a counter-proposal (Action K:*buildCounterProposal*) and sends it to the contractee agent (Action L:*sndMsgCounter-Proposal*). When the contractee agent gets this counter-proposal (*rcvMsgCounterProtocol*), it evaluates this counter-proposal (Action C:*evalCounterProposal*). If the counter-proposal is acceptable and there already are a sufficient number of solutions (a solution is an acceptable commitment with MUG greater than MUC), the negotiation is terminated and the contractee agent informs the contractor agent which commitment is finally built (Action F:*sndMsgFinish*); otherwise, the contractee agent generates a new proposal based on its previous proposal and the current proposal (Action D:*generateNewProposal*), and starts another round of communication.

This mechanism is actually a distributed search process: both agents are trying to find a solution that maximizes the combined utility (the marginal utility gain minus the marginal utility cost). It is not realistic to guarantee an optimal solution given limited computational resources and incomplete knowledge (one agent does not know the other agents' situation), so the goal is to find an acceptable solution, and try to get a better solution if more time is available. The contractee agent first builds an initial proposal which includes the time that the non-local task should be completed and the quality achieved. The time request is a time range defined by the earliest possible time the non-local task NL can start and latest reasonable time the non-local task can be finished. Since there are sequencing requirements and interrelationships among tasks, there may be some task that must be finished before the non-local task can start, and there may be some other task that can't start before the non-local task is finished. For the non-local task, the earliest possible start time is the earliest possible finish time for those tasks (pretasks) that have to be finished before the non-local task can start, the latest finish time is the latest start time for those tasks (without violating their deadline) that have to be performed after the non-local task is finished. The contractee agent gets maximum marginal utility gain during this time range, and the gain is indifferent to when the non-local task is actually executed during this range. Outside of this range, the marginal utility gain decreases, but it may still be worthwhile to search because the marginal utility cost for the other agent may also decrease. So each subsequent proposal from the contractee is built from its own previous proposal by moving the time request

[4]To compute the marginal utility cost in a real time system, not only the actual usage of resource should be considered, but also the opportunity cost may be involved: when the contractor agent makes a commitment to the task NL, it loses the opportunity to perform another incoming task with higher utility, and thus the marginal utility cost may be higher than the immediate utility decrease from performing this task. Similarly when the contractee agent contracts the task NL out, it leaves itself more freedom to accept another higher utility task, hence the marginal utility gain may be higher than the immediate utility increment from the task NL. In this work, the opportunity cost is not addressed in the calculation of marginal utility cost.

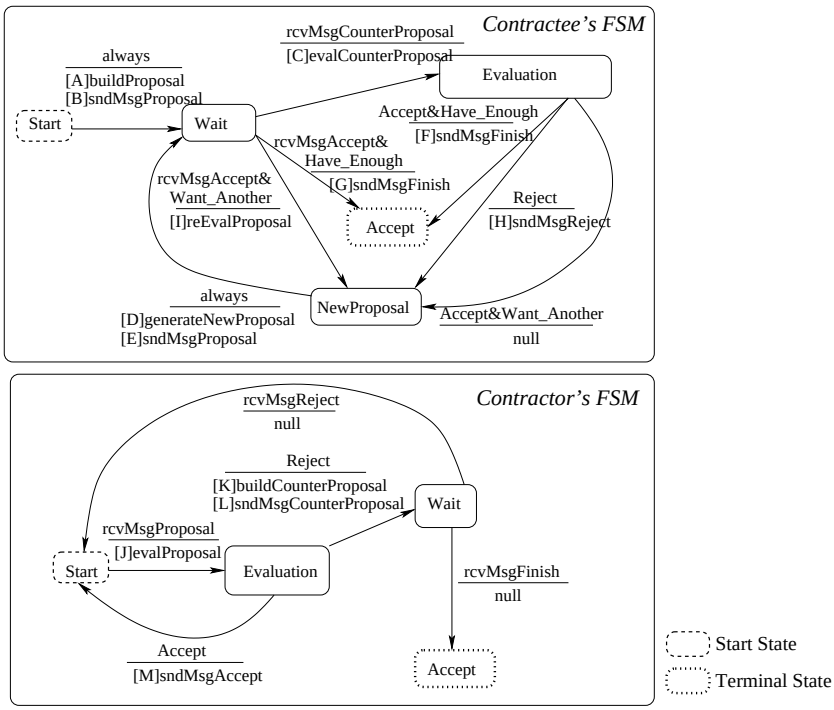

Figure 1: Cooperative task allocation protocol

later. The mechanism also allows for the possibility of varying task NL's quality throughout the range specified by the way that the contractor can accomplish the task. In this way, through additional search on these alternative time ranges, the negotiation process has an anytime character where additional time increases the likelihood of getting a better solution. It is assumed that the environmental situation does not change, such as the arrival of new tasks and the change of current commitments, or the agent does not consider these changes during the negotiation process.

Let us describe the mechanism in greater detail. This protocol uses three functions. One generates an initial proposal by the contractee (**buildProposal**), the second generates a counter-proposal by the contractor (**CounterProposalGeneration**), and the third has the contractee generate a new proposal in response to the counter-proposal (**NewProposalGeneration**). When the contractee obtains its task structure and finds that there is a non-local task NL which needs to be assigned to another agent, it builds a proposal commitment PC based on its local schedule. This commitment specifies the earliest start time, the latest finish time and the quality request for task NL's execution (buildProposal function, see section 2.3). In addition to this information, the marginal utility gain of this commitment is also provided by the contractee. This commitment and associated information are sent to the contractor. The contractor evaluates this commitment in the context of its existing set of potential activities and other commitments as specified in its local task structure by asking a "what-if" question to the scheduler. If this commitment can be satisfied with the marginal utility gain greater than the marginal utility cost, the contractor accepts this commitment; otherwise, the contractor tries to refine this commitment (CounterProposalGeneration function, see section 2.3), and sends a counter-proposal CC to the contractee. When the contractee receives this counter-proposal CC, it evaluates CC by adding it to its local task structure and evaluating the resulting local schedule. If there is a local schedule whose marginal utility gain exceeds the marginal utility cost of the counter-proposal, the counter-proposal is accepted; otherwise, it is rejected. If the counter-proposal is rejected or the

contractee wants to find a better commitment, the contractee tries to improve the commitment (NewProposalGeneration function, see section 2.3). The improvement is a two-dimensional search process based on the time and quality requirement suggested in the previous commitment and counter-proposal from the contractor. The new commitment is sent to the contractor and another negotiation cycle starts. As the negotiation progresses, the contractee keeps track of the number of accepted commitments and stores the accepted commitment with the highest global utility. The negotiation process ends when either the number of negotiation cycles exceeds a predefined limit or the contractee has recognized that the desirable number of improvements over the original accepted commitment has been made. If the contractee has found an acceptable commitment by the time when any of these events occurred, the contractee notifies the contractor of the commitment that has been finally agreed upon.

The mechanism described above also can be applied to multiple contractor agents. The contractee agent can start concurrent negotiation processes with each of the contractor agents, and pick the best acceptable commitment generated.

2.3 Elaboration of Protocol Functions

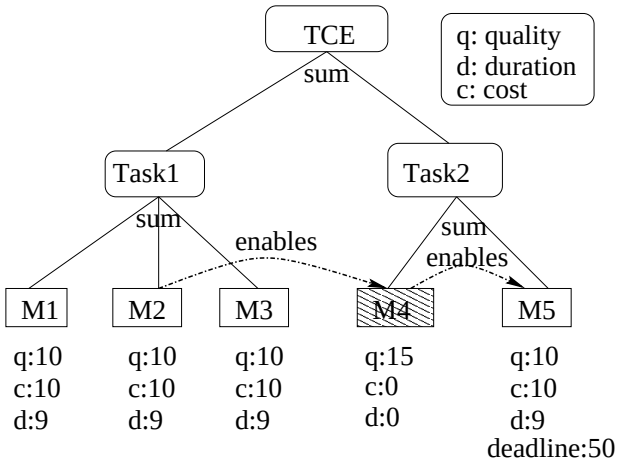

Figure 2: The contractee's task structure

Let us introduce an example that will be used to explain how the following functions work. There is a contractee agent working on task TCE (Figure 2). TCE has two subtasks, Task1 and Task2. Task1 has three subtasks, M1, M2 and M3. Each of them takes 9 units of processing time (d:9), has a cost of 10 (c:10) and generates 10 units of quality (q:10). The "sum" associated with a task means the quality of the task is the sum of all its subtasks. Task2 has two subtasks, M4 and M5. There is an "enables" relationship between M2 and M4, which denotes that M4 can only be started after M2 has been successfully finished. Likewise, another "enables" relationship between M4 and M5 specifies that M5 has to be performed after M4. The deadline constraint associated with M5 indicates it has to be finished by time 50. Subtask M4 is a task that needs to be assigned to another agent (suppose the problem solver makes this decision). Since M4 is not executed by the contractee agent, it does not take local processing time, its duration is set to 0 ("looks like" 0) for the contractee agent.

The **buildProposal** function is used by the contractee agent to build an initial proposal PC. When the contractee finds out that there is a non-local task NL that needs to be assigned to another agent, it first performs a local scheduling process, which assumes the non-local task can be executed by the contractor agent at any time. As a result the contractee gets its local best schedule with the highest local utility achieved. For example, the best local schedule found for task TCE is: $S2 : M2(0-9)M3(9-$

$18)M1(18-27)M4[27-27]M5(27-36)$. It analyzes this schedule and finds the earliest start time and the latest finish time for the non-local task required by the tasks related to this non-local task. The earliest start time and the latest finish time define a range that maximizes the marginal quality gain. The length of this range is dependent on the relationships between the non-local task and other tasks, as well as the time constraints on other tasks. In this example, given the "enables" relationships from M2 to M4, and from M4 to M5, the time range found for M4 is [9, 27]. Besides this time range, this initial proposal also specifies the quality request for NL's execution. The contractee agent doesn't know exactly the specific task qualities that can be achieved and how long it takes or how much it costs to achieve a certain quality. The contractee only knows the range of values that task NL's quality can take, and the estimated duration of task NL. The decision about what quality to choose is important because if the initial quality request is too high, the contractor agent may fail to achieve it given the time range constraint, or even if it is achievable, the marginal utility cost may be higher than the gain; hence the proposal fails. On the other hand, if the quality request is too low, it may miss a better solution at this time. A heuristic is used to assign the initial quality request value: if the time range is much longer than the estimated duration of NL (i.e. the time range is larger than one and a half times of the estimated duration), then the quality request is set to a value higher than the average quality value (i.e. 1.2 times the average quality value); if the time range is very short compared to the estimated duration, then the quality request is set to a value lower than the average quality value; otherwise, the quality request is set as the average quality range. Thus the contractee agent will request a higher quality achievement if it is more flexible on time[5]. For example, compared to the estimated duration 10.5, the proposed time range 18 units ([9, 27]) is very flexible. So the contractee agent requires a higher quality achievement (18) given the knowledge of the estimation average quality achievement is 15.

The **CounterProposalGeneration** function is used by the contractor to generate a counter-proposal in response to an unacceptable proposal. The function works as follows. If there is no previous counter-proposal, the contractor builds the first counter-proposal by removing both the time range and the quality request, and finding the schedule that performs task NL with the minimum marginal utility cost among all schedules returned by the DTC scheduler. This counter-proposal has the minimum marginal utility cost because it only respects the contractor agent's constraints and chooses to do the NL task at its most convenient time and in the most convenient way; hence it is more likely to be an acceptable proposal from the perspective of the contractee agent. If a previous counter-proposal exists, the contractor refines the contractee's current proposal by relaxing the time constraints and lowering the quality request alternatively, and this refining process is repeated until an acceptable (MUC $<$ MUG) counter-proposal is found. The algorithm that used to generate a counter-proposal is shown in Appendix Algorithm A.1.

The **NewProposalGeneration** function is used by the contractee to build a new proposal based on the contractee's previous proposal and the contractor's current proposal. If the previous proposal is acceptable for the contractor, the current proposal is actually the contractee's previous proposal with detailed implementation information (such as start time, finish time and quality achievement). If the previous proposal is not acceptable, the current proposal is a counter-proposal from the contractor. The contractee performs a two-dimensional search in the time-quality space[6]. As described before, the initial proposal is built with

[5]Setting the quality request value low does not necessarily result in a speedier search process to find an acceptable solution. A lower value results in the marginal utility cost decrease; however, it also decreases the marginal utility gain. An acceptable solution should have the marginal utility gain greater than the marginal utility cost.

[6]This two-dimensional search has a depth-first search character: for a given range on the time dimension, the search explores all possible values on the quality dimension; afterwards the search is moved to another range on the time dimension. This algorithm also could be generalized for search on more than two dimensions. The assumption is that the values of each dimension are independent.

a time range that maximizes the marginal utility gain. The next new proposal is to search other time areas trying to find a better proposal by reducing marginal utility cost. The initial time range is defined by the earliest start time and the deadline for the NL task. For the non-local task, the earliest start time is the earliest finish time for those tasks (pretasks) that have to be finished before the non-local task can start. The latest finish time is the latest start time for those tasks (without violating their deadline) that have to be performed after the non-local task is finished. The earliest start time can be moved earlier if those pretasks have alternatives that take less time, or part of those pretasks can be dropped without preventing the execution of the NL task[7]. Otherwise, if neither of these two possibilities exist, the earliest start time can't be moved earlier, hence it is unnecessary to search the time area before the initial time range. In the example of Figure 2, the pretask of M4 is M2, which can not be moved earlier for it is already the first element in the schedule. And also there is no alternative way to finish M2 with less time, so the earliest start time for M4 (9) can not be moved earlier. The latest finish time can be moved later, which can result in additional costs being incurred due to the violation of some later tasks' deadlines (hard or soft deadline), which decreases the marginal utility gain. For example, the latest finish time of M4 can be moved later, which causes task M5 to start later and then the whole task TCE to finish later. As long as the deadline is not violated, the task can still produce a valid result. In this work we assume the earliest start time can't be moved earlier and we only search the time area after the initial time range, but the algorithm could easily be adapted to search in both directions. When the initial proposal is built the contractee agent has no idea how long it takes the contractor agent to perform the NL task and how much quality it can achieve. The contractor's current proposal provides this information and it can be used to build a new proposal. How the new proposal is constructed is described as the following.

- If the current quality achievement (qa) is less than the average quality value and the previous proposal is not the initial proposal, the new proposal requests a higher quality and moves the deadline later to make a high-quality performance more likely. The example is shown in Step 10 in Section 3.2.
- If the current quality achievement (qa) is higher than the average quality value and the previous proposal is the initial proposal (remember the initial proposal does not necessarily start with the lowest quality request), the new proposal requests a lower quality with the initial time range to see if a better solution exists with the reduced marginal quality cost. The example is shown in Step 4 in Section 3.2.
- Otherwise, the new proposal moves to a later time range by a step size of 5 (in this example, we choose the step size as 5, which is about a half of the estimated duration of the non-local task, actually the step size can be adjusted)[8], and requests a lower quality trying to reduce the marginal utility cost. The example is shown in Step 7 in Section 3.2.

[7]The reason that some of those pretasks can be dropped without preventing the execution of the NL task is the existence of a "soft relationship". There are two types of relationships: "hard relationship" such as "enables" - task A enables task B means that task B can not generate valid result without the successful execution of task A; "soft relationship", such as "facilitates" - task A enables task B means that the successful execution of task A facilitates the execution of task B in terms of reducing the processing time, cost or increase the quality of task B, if task A is executed before task "B". In this case, task A can be counted as a precondition of task B in the initial computing process of EST for task B. However, task A also can be dropped later in order to leave more room for the negotiation on task "B". The example in this paper does not show the situation of "soft relationship", however, the algorithms described here can be easily modified to handle this situation.

[8]The step size affects the performance of the algorithm in the following way: when the step size is large, it may take less time to find a good solution, but it is also possible to miss some good solutions (for example, when step size is 10, the first range searched is [0, 15], the second range searched should be [10, 25], then the solution that starts at 5 and finishes at 15 will not be found); when the step size is small, it may take longer to find a good solution, but the possibility of missing good solutions is reduced. When the step size is 1, a complete search (in time dimension) is performed.

This new proposal is evaluated and if the gain is larger than the estimated cost (it is a good proposal), it is sent to the contractor; otherwise, the proposal is modified to make it closer to the initial proposal so that the gain could be higher. This process is repeated until a good new proposal is found. The above procedure is applied when the previous proposal is acceptable and the current proposal is actually the contractee's previous proposal with the detailed implementation information. When the previous proposal is not acceptable, the current proposal is a counter-proposal from the contractor. As stated previously, the first counter-proposal is built by throwing away all constraints from the contractee and finding the most convenient way to perform the non-local task. In this situation, the contractee agent analyzes why the previous proposal failed; if it failed because the initial time range was too short, it enlarges the range by moving the deadline later and requests a lower quality to see if there is a solution near the initial proposal. Otherwise it adjusts the initial range to be a little bit longer than the current execution time and requests a quality higher than the average quality. The second counter-proposal and those counter-proposals that follow it are built by relaxing the previous proposal's request and finding a solution as close to the previous proposal as possible. In this situation, the next proposal is built based on the current proposal, by either requesting a higher quality with a later finish time or moving to the next time range by a step size, depending on how much quality is achieved now. The algorithm that used to generate a new proposal is shown in Appendix Algorithm A.2.

2.4 Another Approach - Binary Search

In the previous section we described our algorithm which searches the time dimension range by range, and in each time range, different quality requirements are explored. This algorithm is an approximation of the complete search process; it has a larger search step and uses heuristics to control the search process. Earlier, we tried a binary search algorithm [13] whose short description follows. The contractee builds an initial proposal as described above: this initial proposal requests that the non-local task be performed at the most convenient time for the contractee. If the contractor could not accept this proposal, it builds the first counter-proposal using the same procedure as the one described above. Each next proposal from the contractee is a compromise between its own previous proposal and the contractor's counter-proposal, while each next counter-proposal from the contractor is a compromise between the contractee's proposal and the contractor's own previous counter-proposal. Figure 3 and Figure 4 show how the contractee generates the new proposal based on its previous proposal and the counter-proposal. The contractee agent also behaves differently depending on whether it is trying to improve an existing acceptable commitment or generating a new proposal in response to a rejection. If there is already an acceptable solution, it tries to find a new solution either with a higher MUG or lower MUC, which will hopefully increase the combined utility. It there is no acceptable solution, it tries to find a solution by relaxing previous request constraints (in quality and/or in time).

If there is an existing acceptable commitment, the contractee takes the following actions:

- *The contractor can not do task NL as early as the contractee requested*. The contractee now asks for a finishing time that is the average of those of the counter proposal and the previous proposal. It also decreases the requested quality at a certain rate (by multiplying it by a value "a" between 0 and 1) thus trying to meet the contractor halfway and with a reduced quality (Figure 3, case 1).
- *The contractor can do task NL as the contractee requested.* The contractee asks for a finishing time that is the average of those of the counter proposal and previous proposal and requests the quality that the contractor offered, trying to see if this new pair reduces the contractor's cost and thus increases the combined utility (Figure 3, case 2).

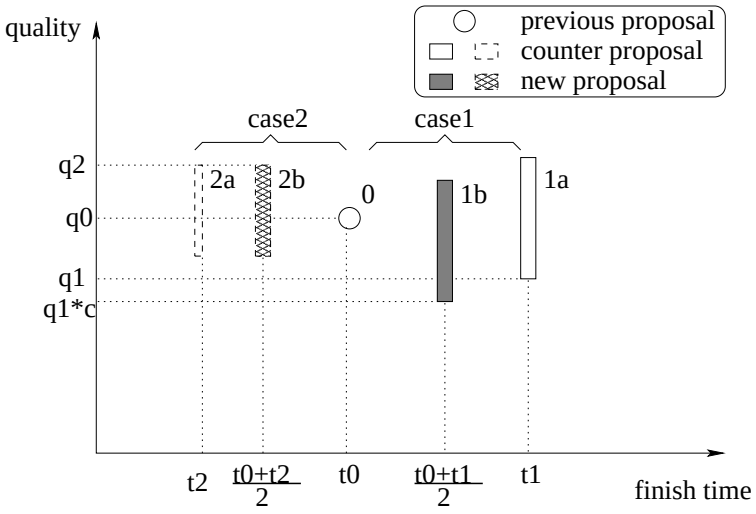

Figure 3: New proposal generation (with acceptable solution)

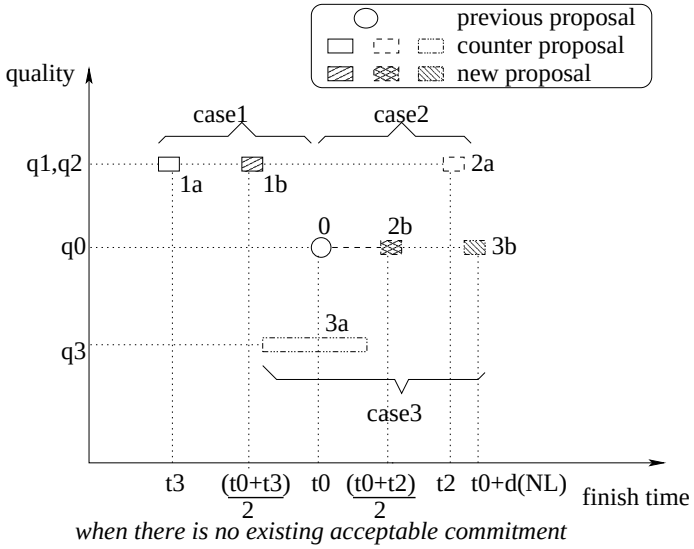

Figure 4: New proposal generation (no acceptable solution)

If there is no existing acceptable commitment yet, the contractee takes the following actions:

- *The contractor can not do task NL as early as the contractee requested, but it can do it later with a higher quality.* The contractee now asks for a finishing time that is the average of those of the counter proposal and previous proposal, and it keeps the requested quality the same, thus trying to meet the contractor halfway (Figure 4, case 2).
- *The contractor can not do task NL as early as the contractee requested and the quality requested is not possible.* The contractee asks for a finishing time that is the sum of that of the previous proposal and the duration estimate of task NL, and it keeps the requested quality the same, thus trying to do the task later (Figure 4, case 3).
- *The contractor can do task NL at the requested time or earlier and even with a higher quality than requested.* The contractee asks for a finishing time that is the average of those of the counter proposal and the previous proposal and requests the quality that the contractor offered, thus trying to see if this new pair reduces the contractor's cost (Figure 4, case 1).
- *The contractor can do task NL at the requested time or earlier but the quality requested is not possible.* The contractee asks for a finishing time that is the sum of that of the previous proposal and the duration estimate of task NL, and it keeps the requested quality the same, thus trying to do the task later (Figure 4, case 3).

This binary search algorithm does not work as well as the range-by-range search because it usually leads to finding fewer acceptable solutions and the quality of those solutions is lower. The following factors contribute to its lower performance:

1. The finish time of the first counter-proposal may not be the actual latest reasonable finish time for the non-local task. The non-local task could be finished later with a higher quality that may provide a higher combined utility. Since the binary search range is restricted by the finish time of the first counter-proposal, any solution later than that will not be found.
2. The negotiation is about multiple issues, such as the earliest start time, deadline, and quality requirement; hence, the midpoint of the two proposals is difficult to guess based on these three issues. In the implementation, we focus on the deadline (and the finish time) while the earliest start time is adjusted according to the deadline.
3. The domain knowledge used to guide the search is incomplete. For example, the duration between the earliest start time and the deadline in the first proposal may be less than the minimum duration of the non-local task's execution by the contractor which causes the failure of the first proposal. When the counter-proposal comes, the minimum duration is available at this time, but it is not used in the rest of the search process.
4. The search process is less structured leading to certain types of solutions often being missed, since both agents are likely to search only in the vicinity of their most favorite proposals.

   Based on the above reasons, we developed the range-by-range search which improves negotiation performance.

3 Negotiation Protocols and Example

3.1 Five Protocols

The negotiation mechanism described in the previous sections serves as a basis for a family of protocol variations differing in the criteria for the negotiation process termination. We examine the following five protocols in this research:

- SingleStep: The contractee sends a proposal commitment (PC) to the contractor, the contractor accepts PC if MUG(PC) $>$ MUC(PC); otherwise it rejects PC, and the negotiation is terminated in failure.

- MultiStep-Multiple(n)-Try: The contractee and the contractor perform the negotiation series - “proposal, counter-proposal, new proposal, ... ” - until ’n’ acceptable solutions with increasing utility gains are found or certain iteration limits are reached (i.e. after 10 proposals has been made). We explore three different values for ’n’ in our experiments which are described next.
  - MultiStep-One-Try: MultiStep-Multiple(n)-Try, n=1;
  - MultiStep-Two-Try: MultiStep-Multiple(n)-Try, n=2;
  - MultiStep-Three-Try: MultiStep-Multiple(n)-Try, n=3;
- MultiStep-Limited-Effort: The contractee and the contractor perform the negotiation series - “proposal, counter-proposal, new proposal, ...” - until certain iteration limits are reached. This protocol explores more possibilities than the above mentioned four protocols when the iteration limit is set to a relatively large number.

Although these protocols differ in the amount of search they do prior to termination, none of them performs a complete search. One reason for that is that generating an optimal local agent schedule for each “what-if” question of the negotiation process is an NP-Hard problem; our scheduler uses heuristics to prune part of the search space and thus not all possible options are expanded. The other reason is that the distributed search space for the possible solutions is also very large and a complete search is too expensive. For example, suppose the earliest start time for the contracted task is 10, the deadline is 30, and the contractor agent has three different approaches to accomplish this task. There would then be a total of 20*3 = 60 possible solutions (starting from time 10, 11, ..., 29 by approach#1, approach#2 or approach#3). And for each possible solution, the agent needs to evaluate it in the context of its other local activities. Thus expending computational effort necessary for a complete search is not feasible. Hence, a range-by-range search (with step size of 5) is performed as an approximation of the complete search.

To examine how different protocols work in different situations and to find out the major factors that affect the outcome of negotiation, we have built two agents: the contractee and the contractor. The utility the agent gains by performing task T using schedule S is a multiple attribute utility function, which is a weighted function of the quality achieved, and the cost and duration expended when performing task T.

$$
\begin{aligned}
utility(S) &= quality\_gain(S) * quality\_weight + \\
&\quad cost\_gain(S) * cost\_weight + \\
&\quad duration\_gain(S) * duration\_weight \\
quality\_gain(S) &= \frac{quality(S)}{quality\_threshold} \\
cost\_gain(S) &= \frac{cost\_limit - cost(S)}{cost\_limit} \\
duration\_gain(S) &= \frac{duration\_limit - duration(S)}{duration\_limit}
\end{aligned}
$$

quality(S), cost(S) and *duration(S)* are the quality achieved, cost spent and time spent by schedule S. *quality_threshold*, *cost_limit*, *duration_limit*, *quality_weight*, *cost_weight* and *duration_weight* are defined in the agent’s criteria function. The first three values specify the quality the agent wants to achieve from this task, the cost and the time it wants to expend on this task; the other three values specify the relative importance of the quality, cost and duration attributes[12].

3.2 Example

In this section, we use an example to explain how the negotiation mechanism works. Consider the situation where the contractee is working on task TCE (Figure 2). Task TCE has been explained in Section 2.3. The contractor is an agent that could potentially perform task M4. (There could be more than one agent with the potential of performing task M4. For clarity we only show one). Similarly, Figure 5 shows the contractor's local task TCR (the left part of the figure).Table 1 Summarizes the following example.

In this example, the contractee has the following criteria definition: $quality\_threshold = 50$, $cost\_limit = 50$, $duration\_limit = 55$, $quality\_weight = 0.7$, $cost\_weight = 0.15$ and $duration\_weight = 0.15$. The contractor has a slightly different set of criteria: $quality\_threshold = 50$, $cost\_limit = 50$, $duration\_limit = 55$, $quality\_weight = 0.7$, $cost\_weight = 0.2$ and $duration\_weight = 0.1$.

Table 1: Negotiation example (CP: Current Proposal; FT: Finish Time; QR: Quality Requested; QA: Quality Achieved; CUI: Combined Utility Increase.)

Step	Agent	Action	CP	FT	QR	QA	MUG	MUC	CUI
1	contractee	Build-Proposal	PC0	27	18		0.295		
2	contractor	Evaluate-Proposal	PC0	24	18	19.5		0.189	
3	contractee	Re-Evaluate-Proposal	PC0	24	18	19.5	0.358	0.189	0.169
4	contractee	Generate-New-Proposal	PC1	27	13.5		0.274		
5	contractor	Evaluate-Proposal	PC1	19	13.5	15		0.163	
6	contractee	Re-Evaluate-Proposal	PC1	19	13.5	15	0.295	0.163	0.132
7	contractee	Generate-New-Proposal	PC2	27	9.0		0.211		
8	contractor	Evaluate-Proposal	PC2	24	9.0	10.5		0.053	
9	contractee	Re-Evaluate-Proposal	PC2	24	9.0	10.5	0.232	0.053	0.179
10	contractee	Generate-New-Proposal	PC3	31	11.55		0.236		
11	contractor	Evaluate-Proposal	PC3	28	11.55	15		0.079	
12	contractee	Re-Evaluate-Proposal	PC3	28	11.55	15	0.293	0.079	0.214

Step 1: Build-Proposal (Action A in Figure 1) The contractee schedules local task structure TCE assuming M4 is not to be done and gets the following schedule S1:

$S1 : M2(0-9)M3(9-18)M1(18-27)$

$Quality(S1) = 30; Cost(S1) = 30; Duration(S1) = 27; Utility(S1) = 0.556$

It schedules TCE assuming that another agent could perform M4 and gets schedule S2:

$S2 : M2(0-9)M3(9-18)M1(18-27)M4[27-27]M5(27-36)$

(with M4's result available at time 27)

$Quality(S2) = 55; Cost(S2) = 40; Duration(S2) = 36; Utility(S2) = 0.8518$

It builds the commitment PC0 based on S2: since M2 enables M4, the earliest start time is 9; the deadline is 27 because it has to be finished before M5's scheduled start time 27; the given time range 18 is very flexible compared to the estimated duration(10.5)[9] , so the quality request is set to a higher value(18.0) rather than the average value(15.0) of the estimation quality

[9]The estimated duration is told by the contractee agent.

achievement.

PC0: [M4, earliest_start_time: 9, latest_finish_time: 27, quality_request: 18]

$MUG(M4) = Utility(S2) - Utility(S1) = 0.8518 - 0.556 = 0.295$

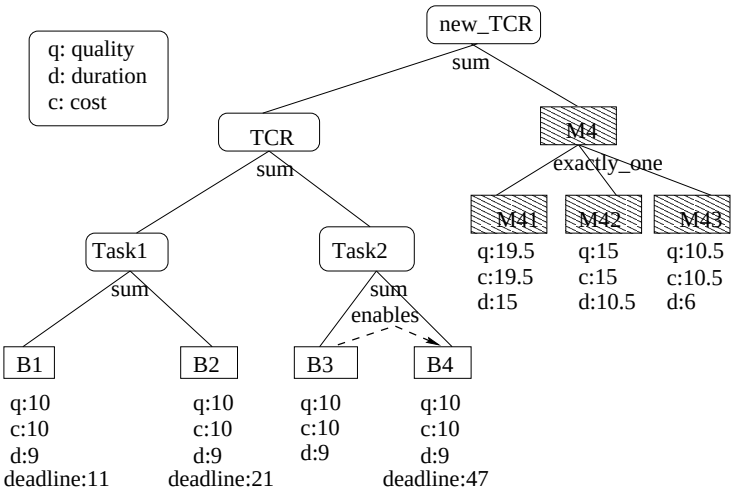

Figure 5: The contractor's task structure

Step 2: Evaluate-Proposal (Action J in Figure 1) The contractor receives this commitment, adds M4 to its local task structure TCR and gets a new task structure new_TCR (Figure 5). The contractor instantiates M4 and finds three different plans to perform M4: M41, M42 and M43. Each plan has different quality, cost and duration characteristics. These three choices are represented as three subtasks of M4 with "exactly_ one" quality accumulation function (qaf) in the TÆMS structure.

The contractor schedules new_TCR with PC0:[M4, earliest_start_time: 9, latest_finish _time: 27, quality_request: 18], and finds the following schedule S3:

$S3 : B2(0-9)M41[9-24]B1(24-33)B4(33-42)$

$Quality^{10}(S3) = 30; Cost(S3) = 49.5; Duration(S3) = 42; Utility(S3) = 0.446$

Compared to the schedule S4 without performing Task M4:

$S4 : B1(0-9)B2(9-18)B3(18-27)B4(27-36)$

$Quality(S4) = 40; Cost(S4) = 40; Duration(S4) = 36; Utility(S4) = 0.635$

the marginal utility cost is Utility(S4) - Utility(S3) = 0.189. Then it sends the following information back to the contractee agent:

PC0 [M4, start_time: 9, finish_time: 24, quality_achieved: 19.5, quality_request: 18]

$MUC(PC0) = Utility(S4) - Utility(S3) = 0.189.$

Step 3: Re-Evaluate-Proposal (Action I in Figure 1) The contractee receives the slightly altered PC0 and re-evaluates it since it has a higher quality and an earlier finish time than the original PC0:

[10]Notice the quality of schedule S3 does not include the quality achieved by M41 since it does not contribute to the contractor's local utility; however, when DTC scheduler works on the task structure "new_TCE", it does count the quality of task M4. The purpose of the calculation without the quality of M41 is to get the marginal utility cost.

PC0 [M4, start_time: 9, finish_time: 24, quality_achieved: 19.5, quality_request: 18]

$MUG(PC0) = 0.358 > MUC(PC0) = 0.189$

This is an acceptable commitment. In either a SingleStep protocol or a MultiStep-One-Try protocol, the contractee stops here and accepts PC0 with the combined utility gain of 0.169. In a MultiStep-Two-Try or a MultiStep-Three-Try protocol, the contractee continues negotiation and tries to find a better commitment.

Step 4: Generate-New-Proposal (Action D in Figure 1) If the contractee decides to find another solution, it attempts to improve the proposal based on its previous proposal and the current proposal from the contractor. It constructs a new proposal by decreasing the quality request, based on Algorithm A.2 in the Appendix:

PC1 [M4, earliest_start_time: 9, latest_finish_time: 27, quality_request: 13.5]

The contractee agent evaluates this new proposal and finds schedule S5 with this commitment.

$S5 : M2(0-9)M3(9-18)M1(18-27)M4[27-27]M5(27-36)$

(with M4's result available at 27 and achieved quality of 13.5)

$Utility(S5) = 0.83, MUG(PC1) = U(S5) - U(S1) = 0.274$

PC1 is sent to the contractor.

Step 5: Evaluate-Proposal (Action J in Figure 1) The contractor finds schedule S6 that satisfies the commitment PC1.

$S6 : B2(0-9)M42[9-19]B3(19-28)B4(28-37)$

$Quality(S6) = 30; Cost(S6) = 45; Duration(S6) = 37; Utility(S6) = 0.472,$

$MUC(PC1) = Utility(S3) - Utility(S6) = 0.635 - 0.472 = 0.163$

Since the marginal gain is greater than the cost, PC1 is acceptable.

Step 6: Re-Evaluate-Proposal (Action I in Figure 1) The contractee receives and re-evaluates the revised PC1 which has a higher quality (15) and an earlier finish time:

PC1 [M4, start_time: 9, finish_time: 19, quality_achieved: 15, quality_request: 13.5]

$S5 : M2(0-9)M3(9-18)M1(18-27)M4[27-27]M5(27-36)$

(with M4's result available at 19 and achieved quality of 15)

$Utility(S5) = 0.851$

$MUG(PC1) = 0.295 > MUC(PC1) = 0.163$

This is also an acceptable commitment. However this commitment with the combined utility gain of 0.132 is worse than the first solution. Thus in a MultiStep-Two-Try protocol or a MultiStep-Three-Try protocol, the contractee continues negotiation and tries to find a better commitment.

Step 7: Generate-New-Proposal (Action D in Figure 1) It rebuilds a new proposal by moving the earliest start time later, from old start time of 9 to 14 by adding 5 (the step size is 5), as specified in Algorithm A.2:

PC2 [M4, earliest_start_time:14, latest_finish_time: 27, quality_request: 9.0]

The contractee agent evaluates this new proposal and finds schedule S7 with this commitment.

$S7 : M2(0-9)M3(9-18)M1(18-27)M4[27-27]M5(27-36)$

(with M4's result available at 27 and achieved quality of 9.0)

$Utility(S7) = 0.767, MUG(PC2) = 0.211$

PC2 is sent to the contractor.

Step 8: Evaluate-Proposal (Action J in Figure 1) The contractor finds schedule S8 that satisfies the commitment PC2.

$S8 : B1(0-9)B2(9-18)M43[18-24]B3(24-33)B4(33-42)$

$Quality(S8) = 40; Cost(S8) = 50.5; Duration(S8) = 42; Utility(S8) = 0.582,$

$MUC(PC2) = Utility(S3) - Utility(S8) = 0.635 - 0.582 = 0.053$

Since the marginal gain is greater than the cost, PC2 is acceptable.

Step 9: Re-Evaluate-Proposal (Action I in Figure 1) The contractee receives PC2 and re-evaluates it based on the higher quality and the earlier than requested finish time it gets:

PC2 [M4, start_time:18, finish_time: 24, quality_achieved: 10.5, quality_request: 9.0]

$MUG(PC2) = 0.232 > MUC(PC2) = 0.053$

This is a better acceptable commitment than previously generated. In a MultiStep-Two-Try protocol, the contractee agent will stop and accept this commitment with the combined utility gain of 0.179. In a MultiStep-Three-Try protocol, the contractee continues negotiation to find a better commitment.

Step 10: Generate-New-Proposal (Action D in Figure 1) It rebuilds a new proposal by requesting a higher quality and extending the deadline, as Algorithm A.2 describes:

PC3 [M4, earliest_start_time:18, latest_finish_time: 31, quality_request: 11.55]

The contractee agent evaluates this new proposal and finds schedule S9 with this commitment.

$S9 : M2(0-9)M3(9-18)M1(18-27)M4[31-31]M5(31-40)$

(with M4's result available at 31 and achieved quality of 11.55)

$Utility(S9) = 0.767, MUG(PC3) = 0.236$

PC3 is sent to the contractor.

Step 11: Evaluate-Proposal (Action J in Figure 1) The contractor finds schedule S10 that satisfies the commitment PC3.

$S10 : B1(0-9)B2(9-18)M42[18-28]B3(28-37)B4(37-46)$

$Quality(S10) = 40; Cost(S10) = 55; Duration(S10) = 46; Utility(S10) = 0.556$

$MUC(PC2) = Utility(S3) - Utility(S10) = 0.635 - 0.556 = 0.079$

Since the marginal gain is greater than the cost, PC3 is acceptable.

Step 12: Re-Evaluate-Proposal (Action I in Figure 1) The contractee receives PC3 and re-evaluates it based on the higher quality and the earlier than requested finish time it gets:

PC3 [M4, start_time:18, finish_time: 28, quality_achieved: 15, quality_request: 11.55]

$MUG(PC3) = 0.293 > MUC(PC3) = 0.079$

Thus PC3 with a combined utility gain of 0.214 is the best solution found so far. In a MultiStep-Three-Try protocol, the contractee agent will accept this commitment and stop; in a MultiStep-Limited-Search protocol, if the predefined iteration limits have not been reached, the agent will continue searching.

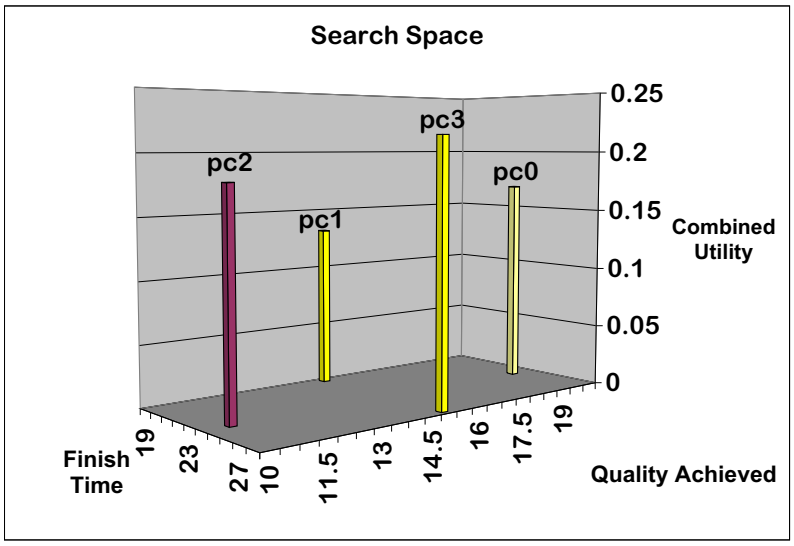

Figure 6: Search space

As a result of this negotiation, the contractee has obtained four acceptable commitments: PC0 starts at 9 and finishes at 24, achieves quality 19.5 and has a combined utility increment of 0.169, PC1 starts at 9 and finishes at 19, achieves quality 15 and has a combined utility increment of 0.132, PC2 starts at 18 and finishes at 24, achieves quality 10.5 and has a combined utility increment of 0.179, PC3 starts at 18 and finishes at 28, achieves quality 15 and has a combined utility increment of 0.214. PC3 is the best solution. Figure 6 shows PC, PC1, PC2 and PC3 in the 2-dimensional search space.

4 Experiment & Evaluation

The experiments are designed to examine how negotiation protocols with different stopping criteria perform in different situations and find what are the major factors that affect the performance. Two agents have been constructed, the contractee agent and the contractor agent. Each agent sequentially processes a set of different task structures. Each task structure is generated as a variant of the basic task structure shown in Figure 2 and Figure 5. Although the basic task structure does not change in this experiments, it represents a set of problems where the negotiation occurs over a non-local task which has interrelationships with other local tasks, and both the contractee agent and the contractor agent deal with complex local tasks which carry temporal constraints and interrelationships among them. The number of temporal constraints (deadline and earliest start time) attached to a task varies from 0 to 3, and the number of "enables" interrelationships among tasks varies from 0 to 3. For example, in Figure 2, there is a deadline constraint attached to task M2, and there is an "enables" interrelationship between M4 and M5. The purpose is to generate negotiation contexts with different degrees of difficulty. There is a total of 4096 ($2^6 * 2^6$) test cases resulting from the combinations of these task structures[11]. Figure 7 shows the contractee's task structure and the contractor's task structure with all

[11]We recognized the limitation of this experimental setup. There are two ways to set up the experiment. One is to generate all task structures randomly; the other is to use a template task structure and vary it. We choose the second approach. The reason is that the first approach brings

six possible temporal constraints and all six possible interrelationships. Besides the five different protocols described in section 3.1, we also developed a "near-optimal-search" algorithm as a comparison base for the experiment. The "near-optimal-search" algorithm searches each start time point and finish time point in a reasonable time range, combined with each possible approach for the non-local task. This "near-optimal-search" however is still not guaranteed to find the optimal solution since the scheduler we use is itself heuristic and does not always find the optimal local schedule for the given constraints. Although the local scheduling is still not "complete", both agents explore all generated possibilities and find which solution has the highest combined utility; such a solution is called the "best-found solution". We compare the solution from each protocol to the best-found solution to evaluate its effectiveness.

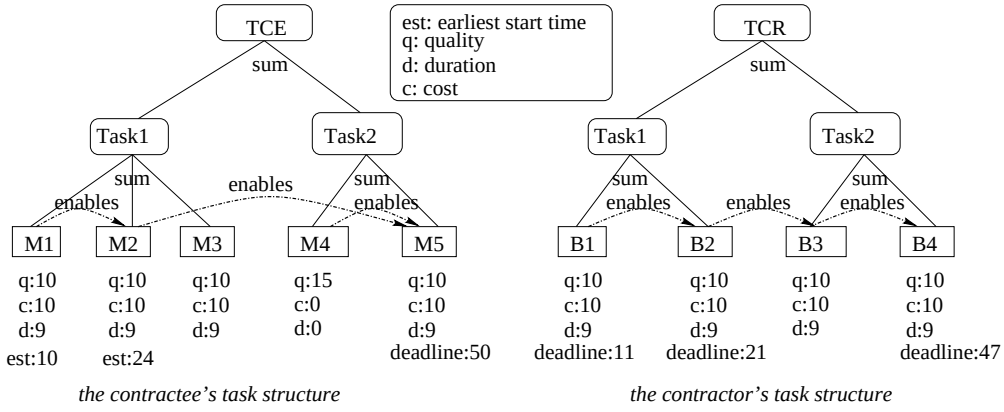

Figure 7: Examples of various task structures

We collected the following data for each test case in the experimental suite:

- *Outcome* (Success/Fail): A negotiation session is successful if it ends with a commitment that increases the combined utility. Otherwise it fails.
- *Utility Gain*: The difference between the MUG(C) and MUC(C). C is the finally adopted commitment. If the negotiation session fails, Utility Gain is 0.
- *Gain Percentage*: The percentage of the utility gain with respect to the combined utility achieved without performing the task allocation.
- *Solution Quality*: The goodness of the solution compared to the best-found solution from the "near-optimal-search". We compare only the utility increase as a result of the negotiation. Suppose a negotiation solution results in the combined utility increased of 18%, and the best-found negotiation solution would increase the combined utility by 20%, then the quality of this negotiation solution is 90% (= 18/20*100%). If a negotiation fails without reaching an agreement, the quality of the solution is defined as 0.
- *Complexity of Task Structures*: A complexity measure of the negotiation is calculated based on the number of constraints

too many variations, hence requiring a large amount of test cases which is not computationally feasible in our situation. Another reason that we choose the second approach is, by using the same template, it is easier to analyze the data and understand what are important characteristics that affect the performance of the algorithms. The experimental results are not intended to be definitive but rather to provide a road map for agent designers to think about complex negotiation protocols and their effectiveness under different situations.

("deadline" and "enables" relationships) in the task structures. The formula we use to calculate this complexity measure is as follows:

$complexity = ir1 + tc1 + ir2 + tc2 + \frac{ir1*tc1+ir2*tc2+ir1*ir2+tc1*tc2+ir1*tc2+ir2*tc1}{6}$

$ir1$: number of interrelationships in the contractee's task structure; $tc1$: number of temporal constraints in the contractee's task structure; $ir2$: number of interrelationships in the contractor's task structure; $tc2$: number of temporal constraints in the contractor's task structure;

For example, in Figure 7, $ir1 = 3, tc1 = 3, ir2 = 3, tc2 = 3, complexity = 21$. This formula is based on the idea that the more constraints there are, the more complicated the task structures are, and the more difficult the negotiation should be. The range of the complexity function in this experimental suite is [0, 21].

- *Number of Negotiation Steps*: The length of the negotiation series (Proposal[1] - Counter-Proposal[2]- Proposal[3] - Counter-Proposal[4] - ...).

Table 2: Comparison of five protocols (AGP: the average of the gain percentage over all 4096 cases. ANNS: the average number of the negotiation steps over all the cases. GPS: the negotiation gain per negotiation step (GPS=AGP/ANNS). SQ: the average solution quality over all 4096 cases. AGPS: the average of the gain percentage over all successful cases. SQS: the average of the solution quality over all successful cases.)

	Success	AGP	ANNS	GPS	SQ	AGPS	SQS
SingleStep	2580	7.63	1.0	7.63	51.44	12.11	81.64
MultiStep-One-Try	4088	10.17	1.48	6.87	72.37	10.19	72.51
MultiStep-Two-Try	4088	11.9	4.69	2.55	84.57	11.97	84.74
MultiStep-Three-Try	4088	13.4	6.42	2.09	96.21	13.43	96.39
MultiStep-Limited-Effort	4088	13.9	8.15	1.7	99.36	13.93	99.55

Table 2 shows the comparison of these five protocols. Out of the 4096 test cases, the SingleStep protocol succeeds in 2580 cases, the other four MultiStep protocols succeed in 4088 cases. Among these 4088 cases, there are 1508 cases in which the MultiStep-One-Try protocol finds a better solution than the SingleStep protocol; there are 2298 cases in which the MultiStep-Two-Try protocol finds a better solution than the MultiStep-One-Try protocol; there are 2168 cases in which the MultiStep-Three-Try protocol finds a better solution than the MultiStep-Two-Try protocol; there are 675 cases in which the MultiStep-Limited-Effort protocol finds a better solution than the MultiStep-Three-Try protocol. For the SingleStep protocol, the average solution quality(SQ) is 51.44% of the best-found solution; the average number of the negotiation steps(ANNS) is 1, and the average utility gain from negotiation (AGP) is 7.63% of the combined utility without negotiation. Hence the average negotiation gain over each negotiation step (GPS=AGP/ANNS)) is 7.63% of the combined utility without negotiation. For the four MultiStep protocols, as the average negotiation step number (ANNS) increases from 1.48 to 8.15, the average solution quality(SQ) also increases from 72.37% to 99.36%, while the negotiation gain over each step (GPS) decreases from 6.87% to 1.7%.

Figure 8 shows how these five protocols perform based on our simple measure of negotiation complexity. As the complexity of the task structures increases, the negotiation problem becomes harder to solve, because the search space for a potentially valid solution is narrowed as the number of constraints in the task structures grows. The SingleStep protocol performs almost as well as the other four protocols when the problem is very easy (the complexity is very low), but its performance decreases dramatically

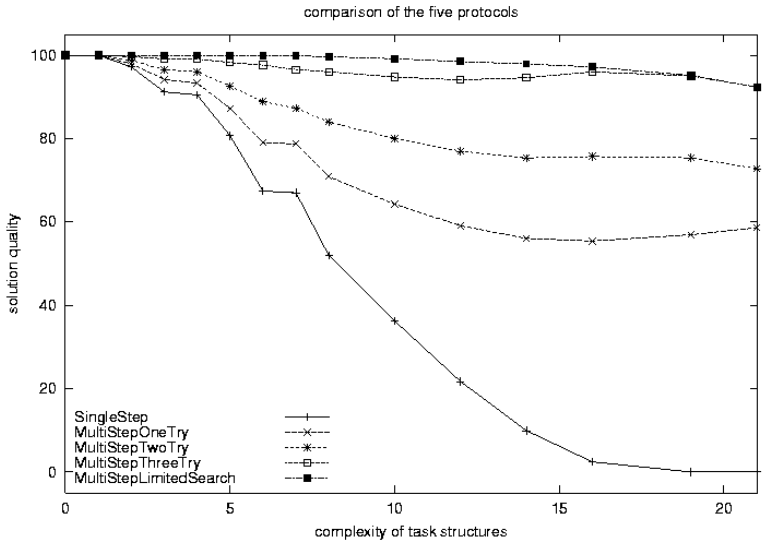

Figure 8: Comparison of five protocols according to the task structure complexity measure(the solution quality is a relative quality compared to the best-found solution, number 100 means the best-found solution)

as the complexity increases. Furthermore, Figure 8 tells us that the MultiStep-$(n+1)$-Try protocol performs much better than the MultiStep-n-Try protocol in the more constrained situation (e.g. when complexity is larger than 5). When there are fewer constraints or too many constraints, the extra search beyond the MultiStep-Three-Try does not bring additional gains. This is because when there are fewer constraints it is very likely that the previous search has found a very good solution; and when there are many constraints, it is hard to find a better solution as a result of the extra search.

Figure 9 shows the performance of each protocol with error bar (the confidence level is 0.9). We find that as the negotiation effort increases, the performance of the protocol is more stable. SingleStep protocol sometimes fails even in the medium complexity situation, MultiStep-One-Try protocol sometimes has a solution quality that is only 10% of the best-found solution. However, with MultiStep-Two-Try protocol, we have the confidence that over 90% of the time, the solution quality is at least 50% of the best-found solution. With MultiStep-Three-Try, the lower bound of the solution quality is raised to 70% of the best-found solution. The agent can choose an appropriate protocol depending on its quality requirement and available time.

The above mentioned data has shown that the performance of each protocol is highly related to the difficulty of the specific negotiation problem. Because each protocol requires different amounts of negotiation effort, it is important for an agent to choose an appropriate protocol that balances the negotiation gain and negotiation effort. Negotiation gain can be represented as the *utility gain* from the negotiation; negotiation effort can be measured by the *number of negotiation steps*. The negotiation effort grows as the *number of negotiation steps* increases. The negotiation cost affects the agent's utility for the following reasons. The first reason is that the negotiation process consumes resources (i.e. time, computational capability, communication capacity, etc.) that otherwise could be used for other tasks; the second reason is that the negotiation process itself has an influence on how and when the contracted task could be executed, which can in some cases reduce the utility. For example, the contracted task

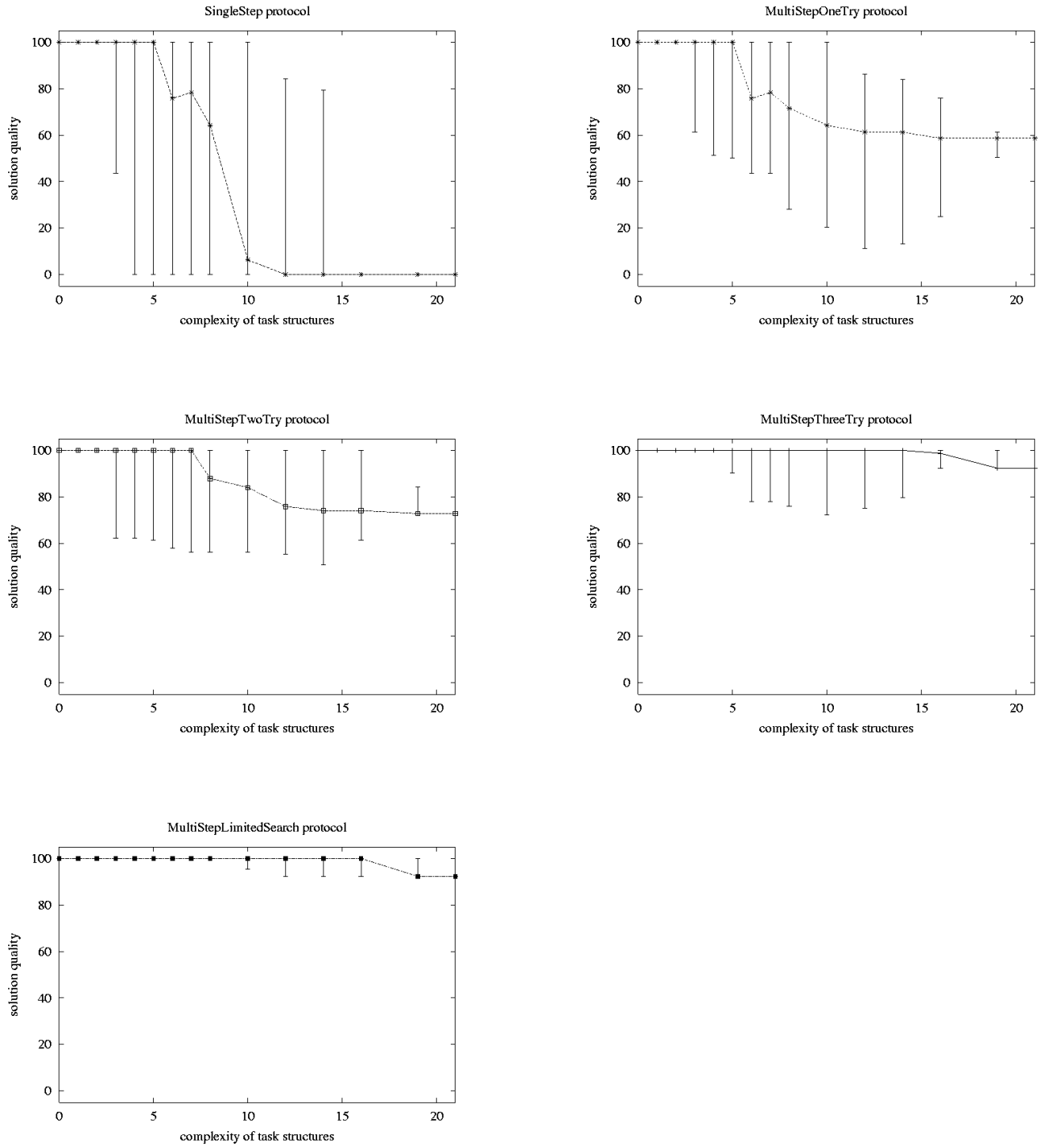

Figure 9: The performance of all protocols with error bars

without negotiation could be started as early as time 10; however the negotiation process also starts at time 10. The longer the negotiation process takes, the later the task can actually be started. More generally, the effect of the negotiation cost on the utility is domain dependent. The following domain characteristics are related: how much slack time there is for the contracted task; how much advance time is available for negotiation without affecting the earliest start time of the task; and the frequency of new tasks arriving, opportunity cost and so forth. Given the above factors, it is hard to measure exactly how the negotiation cost affects the agent's utility. We use the following approximated approach: to make the negotiation cost and gain comparable, the *number of negotiation steps (n)* can be mapped into a certain percentage of utility ($c * n$) by multiplying a constant c. The value of c can be chosen according to the actual situation and it should reflect how the negotiation cost affects the overall utility. Without losing generality, c is set to 0.5 in this experiment. That means each step of negotiation decreases the achieved combined utility by 0.5% the initial combined utility without negotiation. The net negotiation gain in Figure 10 is calculated as the following formula:

$$net\_negotiation\_gain = \frac{\frac{U_n - U_0}{U_0} - c*n}{\frac{U_{best} - U_0}{U_0}}$$

U_n: combined utility after negotiation; U_0: initial combined utility without negotiation; U_{best}: combined utility with the best negotiation solution.

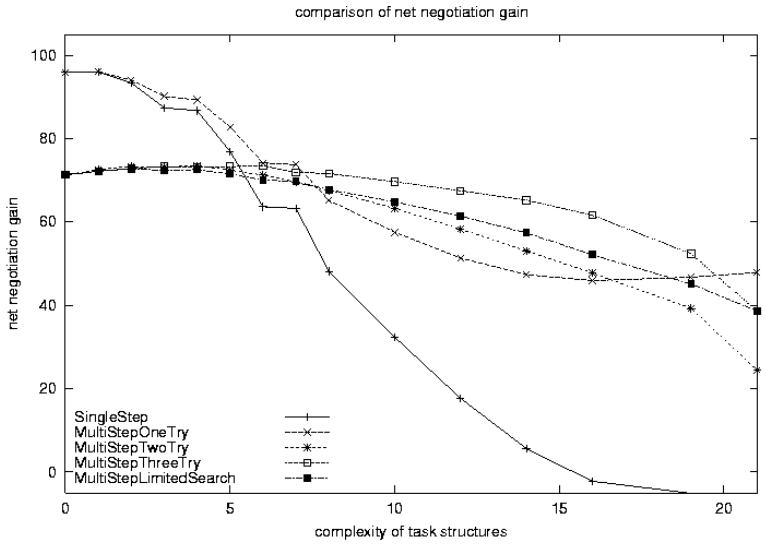

Figure 10: Negotiation gain beyond effort

Figure 10 shows the comparison of the *net negotiation gain* (the negotiation gain minus the negotiation effort) of the five protocols. In Figure 10, we can see a phase transition like phenomenon[3]: when the negotiation situation is very simple (complexity < 5), the Single-Step protocol works as well as the MultiStep-One-Try protocol, and the MultiStep-Two-Try protocol and the MultiStep-Three-Try protocol are not good choices. When the negotiation situation is very difficult (complexity > 19), the MultiStep-One-Try protocol should be chosen; the extra negotiation effort of the MultiStep-Two-Try and the MultiStep-Three-Try protocol does not bring reasonable extra gain. When the negotiation situation is of medium difficulty, then the extra gain exceeds

the extra effort, and the MultiStep-Three-Try protocol is advantageous in this phase. The MultiStep-Limited-Effort is not a good choice in this experimental setup since the negotiation cost is too high. The difficulty of the negotiation situation is simply measured by the number of constraints in the agents' task structures, which is a "reasonable" measure but by no means a completely accurate measure of the distributed search complexity. Though this measure is very simple, it does provide important predicative information. Thus, it seems appropriate for the contractor agent to inform the contractee agent of its local constraints number before the negotiation process starts; the contractee agent can then decide which protocol to use (how much effort to put in the negotiation) according to the estimate of the negotiation difficulty.

complexity	< 5.0			5.0 – 19.0				>=19.0		
	AGP	ANNS	GPS	AGP	ANNS	GPS		AGP	ANNS	GPS
SingleStep	12.81	1	12.81	7.24	1	7.24		0	1	0
MultiStep-One-Try	13.07	1.05	12.45	9.96	1.51	6.59		5.8	2	2.9
MultiStep-Two-Try	13.3	6.14	2.16	11.85	4.57	2.6		7.89	7.23	1.09
MultiStep-Three-Try	13.58	6.83	2.2	13.4	6.38	2.1		9.73	8.62	1.13
MultiStep-Limited-Effort	13.67	7.15	2.22	13.93	8.23	1.69		9.74	9.77	0.997
utility/negotiation-step	< 0.20	< 5.19	> 5.19	< 0.29	< 0.78	< 5.31	> 5.31	< 0.39	< 5.46	> 5.46
SingleStep			best				best			best
MultiStep-One-Try		best				best			best	
MultiStep-Two-Try										
MultiStep-Three-Try					best					
MultiStep-Limited-Effort	best			best				best		

Figure 11: Comparison of five protocols (AGP: the average of the gain percentage. ANNS: the average number of the negotiation steps. GPS: the negotiation gain over each step.)

Figure 11 shows each protocol's performance under the three different situations: low complexity, medium complexity and high complexity task structures. The table shows the amount of negotiation gain (AGP) and negotiation cost (ANNS). The negotiation gain over each step (GPS) provides an upper bound limit on the usefulness of the protocol (cost is less than gain). Let us assume that in a specified application domain, each negotiation step costs c percent of overall utility, then if c is less than GPS the protocol is useful. For example, in the low complexity situation, the MultiStep-Two-Try protocol could be useful only if each negotiation step costs less than 2.16% of overall utility. Also the table shows under each of those three situations, which protocol is the best one when the negotiation cost changes. For example, in the situation of medium complexity (complexity between 5 and 19), when each negotiation step costs less than 0.29% of overall utility, the MultiStep-Limited-Effort protocol is the best; when each negotiation step costs more than 0.29% but less than 0.78% of overall utility, the MultiStep-Three-Try protocol is the best; when each negotiation step costs more than 0.78% but less than 5.31% of overall utility, the MultiStep-One-Try protocol is the best; when each negotiation step costs more than 5.31% of overall utility, the SingleStep protocol is the best. Figure 12 illustrates the above information in another way.

Based on these empirical results, the following observations can be made:

1. In almost all the situations (except the very simple situation), the MultiStep-One-Try protocol is much better than the SingleStep protocol, since it achieves significantly more gain with little extra effort.
2. The MultiStep-Two-Try and MultiStep-Three-Try protocols are worthwhile in the medium-difficult negotiation situation. The agent could decide if it is worthwhile to spend any extra effort. If the task structures have very few or very tight constraints then the MultiStep-One-Try protocol is sufficient.
3. The complexity measure based on the number of constraints can be used to choose the appropriate protocol that balances the

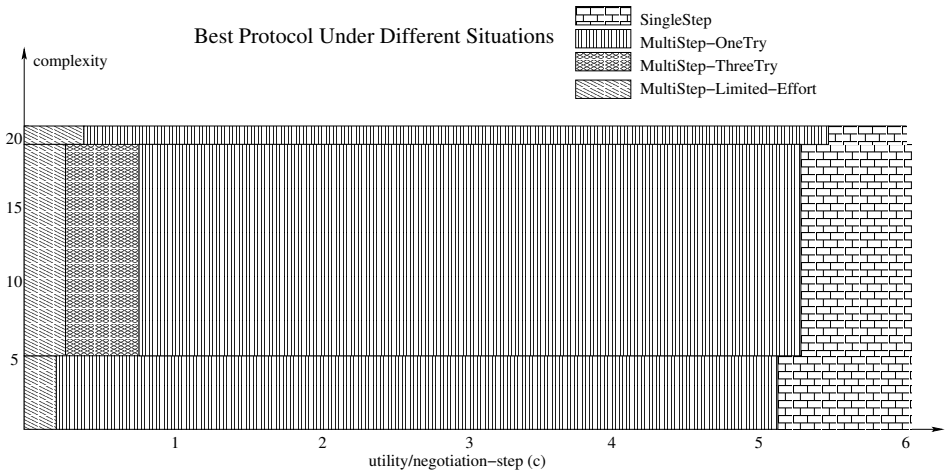

Figure 12: Best protocol under different situations

negotiation gain and effort.

5 Additional Thoughts

As mentioned in the previous section, the protocols' performance is highly related to the difficulty of the negotiation problem; we tried to find some good predictors that the agent can use to select the best protocol. The complexity measure is such a predictor; it is easily obtained (just count the number of constraints) and it works well, as shown in section 4. However, it is not a perfect measure. The large variance in the data shows that it can't capture more detailed information. Hence, we have done additional work trying to find a better predictor.

5.1 Analysis of Solution Space

First, we tried to understand what makes a negotiation problem difficult. The structure of the solution space was examined because we thought the number of solutions may affect the difficulty of a problem. A near-optimal-search was performed and all solutions were obtained (of course, this measure could not be a predictor, but just helps us analyze the problem). The following four measures were calculated:

1. Solution Number (sn): the number of solutions.
2. Unique Solution Number (usn): the number of unique solutions. This only counts solutions that have different start times.
3. Good Solution Number (gsn): the number of good solutions. If the solution is better than 3/4 of the best-found solution, it is a good solution.
4. Unique Good Solution Number (ugsn): the number of unique, good solutions.

The relationship between the performance of the protocols and these measures was then studied. It needs to be pointed out that the difficulty of the negotiation problem is actually an abstract term. In general, an easy problem should be easy to solve, a hard problem should be hard to solve. However, how hard or how easy to solve a problem also depends on what algorithm is used to solve it. Hence, the performance of the negotiation protocols may not reflect the difficulty of the problem perfectly, but it does give us some sense of the difficulty.

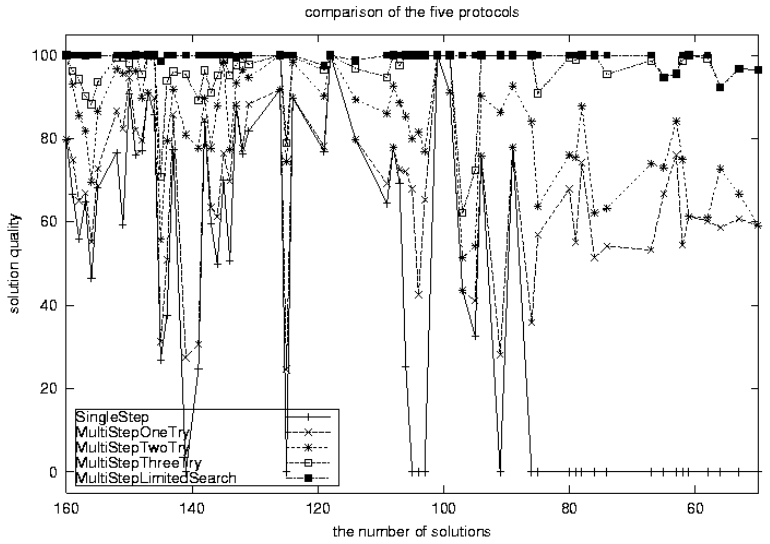

Figure 13: Comparison based on solution number (The solution quality is relative to the best-found solution.)

All four of these measures provide a similar result. Let's use the solution number as an example to explain our observation (Figure 13). It seems that the difficulty is not changing continuously according to the solution number. The solution number divides the problems into two categories: When the solution number is smaller than a certain number (85), the SingleStep protocol always fails, and there is a relatively big gap between the performance of the first two MultiStep protocols and the other two MultiStep protocols. When the solution number is larger than a certain number (85), there is no continuous large gap among all those protocols, although the SingleStep protocol still fails at certain points.

From the above observation we find that it is not only the number of solutions that affects how each protocol performs, but also some other characteristics of the solution space structure, such as how the solutions are distributed in the search space, and how far away the best solution(there may be more than one best solution) is from the initial proposal, etc.

5.2 Definition of Flexibility

We also developed a more complex flexibility measure (extending the complexity measure described by Deshmukh in [6]) that measures how flexible a set of tasks is:

A set of tasks $TS = \{T_1, T_2, \ldots, T_n\}$

For each task T_i: est_i (earliest start time), dl_i (deadline), d_i (duration of processing time)

For any two tasks T_i and T_j: if task T_i must be finished before Task T_j then the precedence function takes value of 1:$p(i,j) = 1$; otherwise $p(i,j) = 0$.

$\varphi_i = \frac{d_i}{\sum_{j=1}^{n} d_j}$

if $p(i,j) = 1$, $\pi_{ij} = dl_i - est_i - d_i$, the slack time of task T_i;

if $p(i,j) = 0$, $\pi_{ij} = 0$;

$\pi^*_{ij} = \pi_{ij} * \varphi_i$

The flexibility of a set of tasks TS is defined as: $F(TS) = -\sum \pi^*_{ij} * \log \pi^*_{ij}$

If there is no feasible linear schedule for the task group, the flexibility is defined as -1.

The flexibility of a single task t is defined as: $F(t) = \frac{dl(t)-est(t)}{d(t)}$.

The characteristic of this flexibility measure is that the more slack time the tasks have, and the fewer precedence relationships among those tasks, the greater the flexibility. If the agent has more slack time for its local tasks and fewer precedence constraints among them, it is easier to arrange its local tasks and hence leave more space for additional tasks. The flexibility measure should capture more detailed information about local tasks compared to the complexity measure.

5.3 Flexibility Related Measures

Based on the flexibility of a set of tasks and the flexibility of a single task, we developed the following measures to describe a negotiation situation: The contractee has a task structure (TSa) and the contractor has a task structure (TSb), and the contractee needs to assign the non-local task Tnl to the contractor.

1. Flexibility Sum Measure (fsum): fsum = F(TSa) + F(TSb)
2. Flexibility Product Measure (fproduct): fproduct = F(TSa)* F(TSb)
3. Flexibility Max Measure (fmax): fmax = Max (F(TSa), F(TSb))
4. Flexibility Min Measure (fmin): fmin = Min (F(TSa), F(TSb))
5. Task Flexibility Sum Measure (tfsum): tfsum = F(Tnl)+F(TSb)
6. Task Flexibility Product Measure (tfproduct): tfproduct = F(Tnl)*F(TSb)
7. Task Flexibility Max Measure (tfmax): tfmax = Max(F(Tnl), F(TSb))
8. Task Flexibility Min Measure (tfmin): tfmin = Min(F(Tnl), F(TSb))

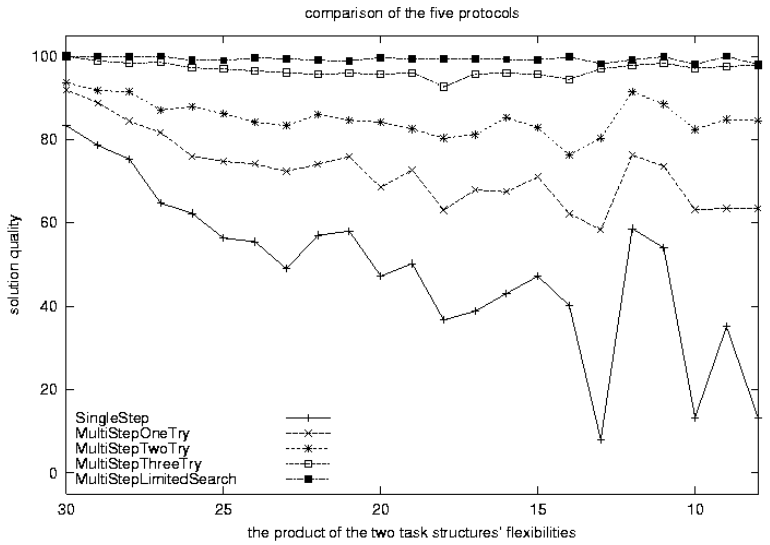

Figure 14: Comparison based on flexibility product measure (fproduct)

Almost all of these measures (except the fmin measure, which does not seem to be a good approach to combine the two flexibility measures) provide results similar to the complexity measure: as the flexibility related measure decreases, the difference among the performance of the protocols increases. Figure 14 shows the comparison of the five protocols according to the $fproduct$ measure.

5.4 Initial Range Related Measures

We also found that the initial proposed range for the non-local task is an important factor that affects the protocols' performance. The following are two measures based on the initial range:

1. Initial Range and Constraints Number Measure ($rconst$): $rconst = (c1 - Initial\_Range(Tnl)/c2) + ir2 + tc2$ [12]
2. Initial Range and Flexibility Measure ($rflex$): $rflex = Initial\_Range(Tnl)/c2 + F(TSb)*c3$

These two measures also provide results similar to the complexity measure. Figure 15 shows the comparison of the five protocols according to the $rconst$ measure.

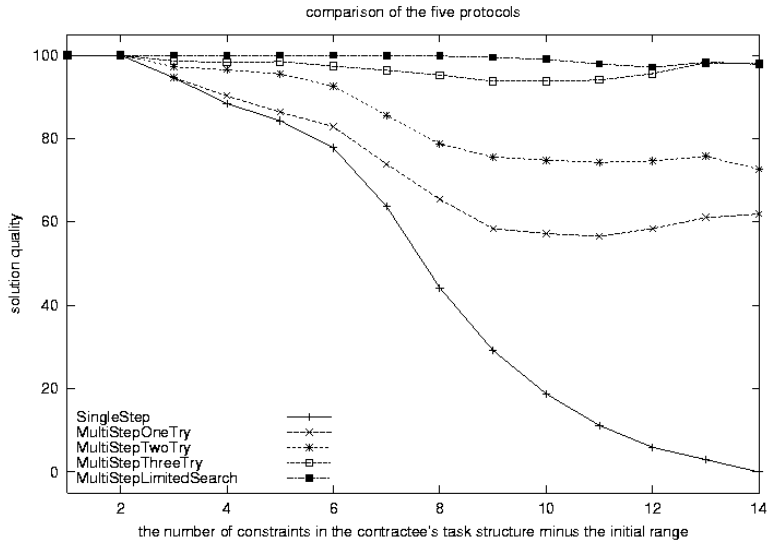

Figure 15: Comparison based on initial range and constraints number measure (rconst)

We have tried all of the above approaches but did not find a completely satisfying measure of the negotiation difficulty. The problem is much harder than we originally thought. The question of how to combine the characteristics of the subproblems together to predict the characteristics of the problem as a whole is still an open issue. In hind sight, given that we are doing a distributed optimization search over interdependent search spaces where there can be complicated interdependence among the spaces, this conclusion is not surprising.

6 Conclusions & Future Work

In this paper, we presented a cooperative, multi-step negotiation mechanism that searches over multiple attributes using a joint utility function that reflects the concerns of both agents in the negotiation. We showed that the application of this mechanism to

[12] ir2: number of interrelationship in TSb; tc2: number of temporal constraints in TSb; c1, c2 and c3 are constant numbers used to adjust the range of the data values. c1=8, c2=5. c3=10.

the task allocation domain in a cooperative system. The contractee agent has a task that needs to be performed by other agents. To perform this task, the contractor agent can choose from several alternatives that produce different qualities and consume different resources. This context requires a complex negotiation that leads to a satisfying solution with increasing combined utility. We first examined a binary search algorithm as a mechanism to find a compromise between the contractee's protocol and the contractor's counter-proposal. After carefully analyzing the trace of this negotiation mechanism, we developed a better way to do the distributed search in the agents' negotiation. The range-by-range algorithm searches a broader space and exploits the domain knowledge from the previous communication to improve the negotiation process. Instead of a tightly constrained proposal, the range proposal allows the contractor agent to have more freedom to effectively react to the requested commitment, which improves the efficiency of the negotiation. The multi-step negotiation mechanism is actually an anytime mechanism: by investing more time, the agent may find a better solution. A set of multi-step protocols are developed based on this mechanism. Experimental work is presented which shows how different protocols work in varying situations. For comparison, a near-optimal-search is performed as a baseline. As a result of this experimental work, we found a phase transition like phenomenon in the operation of the negotiation mechanism: When the negotiation situation is very simple or very difficult, extra negotiation effort does not bring reasonable extra gain. When the negotiation situation is of medium difficulty, the extra gain exceeds the extra effort required by the protocols that do more search. We also found that meta level information could be used to provide advice on how the agent should choose the protocol to balance its gain and effort of negotiation. We develop several predictors to measure the negotiation difficulty, so that an agent can choose the appropriate protocol. This is a very hard problem. Although we have not found a perfect predictor, we did find some simple measures that are helpful.

We would like to continue this work to obtain a better understanding of the negotiation problem characteristics. These characteristics should help us to evaluate the difficulty of a specific negotiation problem better, estimate the probability of finding a good solution before the negotiation is even started, and ultimately, and help the agent make a more reasonable decision about the probable cost and duration of a negotiation, and potential gain.

7 Acknowledgement

We would like to thank Bryan Horling and Regis Vincent for the JAF agent framework and the MASS simulator environment that provided the software infrastructure for the experiments. We also like to thank Abhi Deshmukh for his input on the flexibility measure. We also wish to thank Tom Wagner for the development of the DTC scheduler work and his effort to adopt the scheduler to support this work. We also wish to thank David Jensen for his help with the data analysis.

References

[1] *Foundations of Distributed Artificial Intelligence*, chapter Negotiation Principles. John Wiley and Sons, 1996.

[2] *Multiagent system*, chapter Distributed Rational Decision Making. 1999.

[3] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the really hard problems are. In *Proceedings of IJCAI-91*.

[4] Michael Chia, Daniel Neiman, and Victor Lesser. Coordinating asynchronous agent activities in a distributed scheduling system. In *Proceedings of the Second International Conference on Autonomous Agents (Agents98)*, 1998.

[5] Keith S. Decker and Victor R. Lesser. Quantitative modeling of complex environments. *International Journal of Intelligent Systems in Accounting, Finance, and Management*, 2(4):215–234, December 1993.

[6] Talavage J. J. Deshmukh, A. V. and M. M. Barash. Complexity in manufacturing systems: Part 1 - analysis of static complexity. *IIE Transactions*, 1998.

[7] Laasri H. Lander S. Laasri, B. and V. Lesser. A generic model for intelligent negotiating agents. *International Journal on Intelligent Cooperative Information Systems*, 1992.

[8] S. Lander and V Lesser. Sharing meta-information to guide cooperative search among heterogeneous reusable agents. *IEEE Transactions on Knowledge and Data Engineering*, 1997.

[9] T. Moehlman, V. Lesser, and B. Buteau. Decentralized Negotiation: An Approach to the Distributed Planning Problem. In K. Sycara, editor, *Group Decision and Negotiation*, volume 1, pages 161–192. Kluwer Academic Publishers, 1992.

[10] Tumas Sandholm and Victor Lesser. Issues in automated negotiation and electronic commerece: Extending the contract net framework. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS95)*, 1995.

[11] Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 1998.

[12] Thomas Wagner, Alan Garvey, and Victor Lesser. Criteria-Directed Heuristic Task Scheduling. *International Journal of Approximate Reasoning, Special Issue on Scheduling*, 19(1-2):91–118, 1998.

[13] Xiaoqin Zhang, Rodion Podorozhny, and Victor Lesser. Cooperative, multistep negotiation over a multi-dimensional utility function multi-agent systems negotiation. In *Proceedings of the IASTED International Conference, Artificial Intelligence and Soft Computing (ASC 2000)*.

A Algorithms

Algorithm A.1 Refining process
Related variables: *current proposal (CP): est (earliest start time), dl (deadline), min1 (quality request)*
delt_t1 (=2), delt_t2 (=3): a short period of time;
reduce_ratio (=0.6): a small number ($0.0 < reduce\_ratio < 1.0$*) used to reduce the minimum quality request of current proposal;*
begin
n=0;
repeat
n++;
if ((n mod 2) == 1)
//relax the time constraints
est = est - delt_t1;
dl = dl + delt_t2;
else
//lower the quality request
*minq = minq * reduce_ratio;*
schedules local tasks and NL with new requests (est, dl, minq);
if a schedule contains NL with all requests satisfied and MUC < MUG
// found an acceptable counter-proposal
build the new counter-proposal (NCP) based on this schedule
(the start time (st) and the finish time (ft) for NL and NL's quality achievement
are extracted from the schedule and put into a newly created proposal.)
break;
until a counter-proposal is built
end

Algorithm A.2 New proposal generating process
Related variables: *Initial proposal (IP): est0 (earliest start time), dl0 (deadline), minq0 (quality request);*
Previous proposal (PP): est1 (earliest start time), dl1 (deadline), minq1 (quality request);
Current proposal (CP): st (start time), ft (finish time), qa (quality achieved);
current duration = ft - st;
muc: marginal utility cost of current proposal;
delt_t (=7): a short period of time;
step_size (=5): the size of the step moved in time dimension;
average_quality_value: the average quality the non-local task may achieve;

quality_increase_ratio (=1.1): a small number ($1.0 < quality\_increase\_ratio < 2.0$*) used to increase the current quality request;*
cost_reduce_ratio (=0.5): a small number ($0.0 < cost\_reduce\_ratio < 1.0$*) used to reduce the current marginal utility cost;*
enlarge_rate (=1.3): a small number ($1.0 < enlarge\_rate < 2.0$*) used to increase current duration;*
quality_reduce_ratio (=0.6): a small number ($0.0 < quality\_reduce\_ratio < 1.0$*) used to reduce the quality request;*

```
begin
    if (PP is acceptable)
        if (qa < average_quality_value and not in the initial range)
            //relax the time constraint and increase the quality requirement
            est = st;
            dl = ft + delt_t;
            minq = average_quality_value * quality_increase_ratio(1.1);
        else if (qa > average_quality_value and in the initial range)
            //lower quality requirement to reduce marginal quality cost
            est = est1;
            dl = dl1;
            minq = average_quality_value * quality_reduce_ratio(0.6);
            muc = muc * cost_reduce_ratio(0.5);
        else
            //move to a later time range
            est = est1 + step_size;
            dl = est + current_duration;
            minq = average_quality_value * quality_reduce_ratio(0.6);
            muc = muc * cost_reduce_ratio(0.5);
    else //previous proposal is rejected
        if (first counter-proposal)
            if(dl1 - est1 < current_duration)
                //enlarge the time range and lower quality requirement
                est = est1;
                dl = est + current_duration * enlarge_rate(1.3);
                minq = average_quality_value * quality_reduce_ratio(0.6);
                muc = muc * cost_reduce_ratio(0.5);
            else
                //adjust the time range to be a little bit longer than the current execution time and request a quality
                //higher than the average quality
                est = est1;
                dl = est + current_duration + delt_t;
                minq = average_quality_value * quality_increse_ratio(1.1);
        else
            if (qa > average_quality_value)
                //request a higher quality with a later finish time
                est = st;
                dl = ft + delt_t;;
                minq = average_quality_value * quality_increase_ratio(1.1);
            else
                //move to the next time range
                est = st + step_size;
                dl = est + current_duration;
                minq = average_quality_value * quality_reduce_ratio(0.6);
                muc = muc * cost_reduce_ratio(0.5);
    repeat
        evaluated new proposal with (est, dl, minq, muc)
        if (mug > muc)
            //find a good new proposal;
            break;
        else
            //move closer to the previous proposal
            if (dl < dl0)
                dl = (dl + dl0)/2;
            else
                dl = est + current_duration + delt_t;
            muc = muc*cost_reduce_rate;
    until a good new proposal is found
end
```