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

XiaoQin Zhang, Victor Lesser, Rodion Podorozhny
Department of Computer Science
University of Massachusetts at Amherst
xqzhang@cs.umass.edu

June 27, 2001

Abstract

We present a multi-dimensional, multistep 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 their 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. 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 the 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 parties first verbalize demands and then move toward an agreement through a process of concession formation or search for new alternatives [5]. In multi-agent systems (MAS), negotiation is used for task and resource allocation, recognition of conflicts, resolution of goal disparities, and determination of the organizational structure, all of these influence the coherence of the agent society.

The negotiation research in multi-agent systems falls into two main categories, competitive negotiation and cooperative negotiation. Competitive negotiation occurs among self-interested agents [8], 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 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.

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 [4]. In fact, we feel there is very little work on cooperative negotiation that explicitly tries to maximize a multi-dimensional global utility function. The closest work to our knowledge is that of Moehlman et al. [6]; however their work involves a much simpler and more structured utility function that avoids quantitative reasoning about the combined utility of the agents. Additionally, their approach is not empirically evaluated in different negotiation situations.

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" [3] 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 puts emphasis on a multi-step negotiation process in which agents engage in a series of proposals and counter offers to decide whether the contractee agent will perform a task for the contractor 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 [7] to structure the search. In that work, these measures were used for a single phase evaluation rather than as a basis for cooperative/distributed search among agents to find the best combined local schedules.

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 they will find a solution that leads to the maximum combined utility; another possibility is 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 they can not find one given a limited search [1].

After the negotiation starts, the 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 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 alternatives chosen above. Therefore, as part of this paper we

[1] Another possibility, which we will not consider in this paper, 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 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.

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

2 Framework -TÆMS & DTC

The TÆMS framework [1] is used to represent the agent's local tasks and activities (See Figure 4). The TÆMS task modeling language is a domain-independent framework used to model the agent's candidate 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.

The cooperative negotiation mechanism makes the assumption that a local planning/scheduling mechanism exists that can decide what method execution actions should take place and when. The local scheduler attempts to maximize a specified multi-dimensional utility function. The DTC (Design-To-Criteria) [9] scheduler is used as the agent's local scheduler in our research. 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.

3 Task Allocation Negotiation Mechanism

In a multi-agent system, an agent may need to contract out one of its local tasks to another agent because it can't perform the task locally. This task can potentially be part of a larger activity that the agent performs in order to achieve some desired goal. 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 alternative way for another 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.

An agent will contract out a task to another agent if it does not have the capabilities to perform this task locally or if it is overloaded. We assume that the agent will use the TÆMS task representation of its activities to 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 the option of doing the task locally or not doing it at all versus the option of contracting will be taken into account.

3.1 Definitions

- Contractor Agent (contractor): the agent which has a task (non-local task NL) that needs to be assigned to another agent, the contractor gains quality from this task when it is completed (TCR is the contractor's local task structure).
- Contractee Agent (contractee): the agent which performs this task for the contractor, it devotes processing time and other resources to this task without directly gaining quality (TCE is the contractee's local task structure).
- Marginal Utility Gain [NL, C] (MUG) - The local utility increment for the contractor by having task NL performed with duration and quality specified as in commitment C.
- Marginal Utility Cost [NL, C] (MUC) - The local utility decrement for the contractee by performing task NL with duration and quality specified as in commitment[2].

3.2 Mechanism

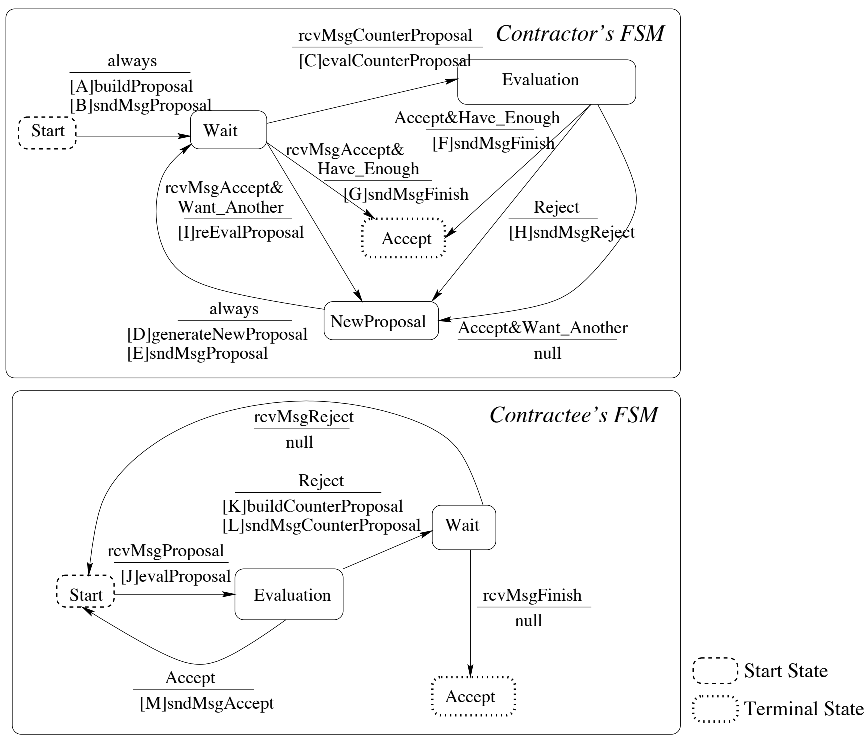

Figure 1: Cooperative task allocation protocol

Figure 1 depicts a Finite State Machine (FSM) model that describes the agents' protocol which implements the task allocation mechanism. The upper part shows the contractor's FSM, the lower part shows the contractee's FSM. The contractor agent starts the negotiation by building a proposal (Action A:*buildProposal*) and sending this proposal (Action B:*sndMsgProposal*) to

[2]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 involved: when the contractee agent make 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 contractor 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.

the contractee agent. After receiving this proposal (*rcvMsgProposal*), the contractee agent evaluates it (Action J:*evalProposal*): if the marginal utility gain is greater than the marginal utility cost, it accepts this proposal (Action M:*sndMsgAccept*); otherwise, this proposal is rejected, the contractee agent builds a counter proposal (Action K:*buildCounterProposal*) and sends it to the contractor agent (Action L:*sndMsgCounterProposal*). When the contractor agent gets this counter proposal (*rcvMsg-CounterProtocol*), 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 contractor agent informs the contractee agent which commitment is finally built (Action F:*sndMsgFinish*); otherwise, the contractor 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 (that is actually to maximize 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 better ones if more time is available. The contractor agent first builds an initial proposal including the time request and the quality request for the non-local task. The time request is a time range defined by the earliest possible time the non-local task can start and latest reasonable time the non-local task NL can be finished. Since there are sequence requirements and interrelationships among tasks, there are some tasks that must be finished before the non-local task can start, and there are some other tasks 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 (preconditions) that have to be finished before the non-local task can start, the latest reasonable 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 contractor 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. The marginal utility gain decreases outside of this range, but it is still worthwhile to search outside of this range because the marginal utility cost for the other agent may also decrease outside of this range. So each subsequent proposal from the contractor is built from its own previous proposal by moving the time request later. The mechanism also allows for the possibility of varying NL's quality throughout the range specified by alternative ways for the contractee to 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.

Let us describe the mechanism in greater detail. This protocol uses three functions. One generates an initial proposal by the contractor, the second generates a counter proposal by the contractee, and the third has the contractor generate a new proposal in response to the counter proposal. When the contractor 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 NL's execution (buildProposal function, see section 3.3). In addition to this information, the marginal utility gain of this commitment is also provided by the contractor. This commitment and associated information are sent to the contractee. The contractee 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 contractee accepts this commitment; otherwise, the contractee tries to refine this commitment (CounterProposalGeneration function, see section 3.3), and sends a counter proposal CC to the contractor. When the contractor receives this counter proposal CC, it evaluates CC by adding it to its local task structure and seeing what the resulting local schedule is. 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 contractor wants to find a better commitment, the contractor tries to improve the commitment (NewProposalGeneration function, see section 3.3). 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 contractee. The new commitment is sent to the contractee and another negotiation cycle starts. As the negotiation progresses, the contractor keeps track of the number of accepted commitments and stores the accepted commitment with the highest global utility. The negotiation process ends either when the number of negotiation cycles exceeds a predefined limit or the contractor has registered that the desirable number of improvements over the original accepted commitment has been made. If the contractor has registered an accepted commitment by the time any of these events occurred, the contractor notifies the contractee of the commitment that has been finally agreed upon.

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

3.3 Elaboration of protocol functions

The **buildProposal** function is used by the contractor agent to build an initial proposal PC. When the contractor 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 contractee agent at any time. As a result the contractor gets its local best schedule with the highest local utility achieved. 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. Besides this time range, this initial proposal also specifies the quality request for NL's execution. The contractor agent doesn't know exactly what different kinds of quality may be achieved and how long it takes or how much it costs to achieve a certain quality. The contractor only knows the range of values that task NL's quality can take, and the estimated duration of the NL. The decision about what quality to choose is important because if the initial quality request is too high, the contractee 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. So the contractor agent requests a higher quality achievement if it is more flexible on time[3].

The **CounterProposalGeneration** function is used by the contractee to generate a counter proposal in response to an unac-

[3]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 to decrease, however, it also decreases the marginal utility gain. An acceptable solution should have the marginal utility gain greater than the marginal utility cost.

ceptable proposal. The function works as follows. If there is no previous counter proposal, the contractee 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. This counter proposal has the minimum marginal utility cost because it only respects the contractee 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. If a previous counter proposal exists, the contractee refines the contractor'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.

Related variables:

current proposal (CC): 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 used to reduce the minimum quality request of current proposal;

Refining process:

```
n=0;
repeat
    n++;
    if ((n mod 2) == 1)
        est = est - delt_t1;
        dl = dl + delt_t2;
    else
        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
        build the new counter proposal (CP) 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
```

The **NewProposalGeneration** function is used by the contractor to build a new proposal based on the contractor's previous proposal and the contractee's current proposal. If the previous proposal is acceptable for the contractee, the current proposal is actually the contractor'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 contractee. The contractor does a two-dimensional search in the time-quality space [4]. As described before, the initial proposal is built with a time

[4] The basic idea of this two-dimensional search is a depth-first search: 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 the the values of each dimension are independent.

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 (preconditions) 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 precondition tasks have alternatives that take less time, or part of those precondition tasks can be dropped without preventing the execution of the NL task. 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. 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. In this paper 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 adopted to search in both directions. When the initial proposal is built the contractor agent has no idea how long it takes the contractee agent to perform the NL task and how much quality it can achieve. The counter proposal provides the information and it can be used to build a new proposal. The following algorithm describes how the new proposal is constructed. If the current quality achievement (qa) is less than the average quality value, the new proposal requests a higher quality and moves the deadline later to make a high-quality performance more likely; 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 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. Otherwise, the new proposal moves to a later time range by a step size of 5 (the step size can be adjusted)[5], which is about a half of the estimated duration of the non-local task, and requests a lower quality trying to reduce the marginal utility cost. 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 contractee; 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 contractor's previous proposal with the detailed implementation information. When the previous proposal is not acceptable, the current proposal is a counter proposal from the contractee. The first counter proposal is built by throwing away all constraints from the contractor and finding the most convenient way to perform the nonlocal task. In this situation, the contractor agent analyzes why the previous proposal fails; if it fails because the initial time range is 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.

[5]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 could 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.

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 nonlocal task may achieve;

quality_increase_ratio (=1.1): a small number used to increase the current quality request;

cost_reduce_ratio (=0.5): a small number used to reduce the current marginal utility cost;

enlarge_rate (=1.3): a small number used to increase current duration;

quality_reduce_ratio (=0.6): a small number used to reduce the quality request;

New proposal generating process:

```
if (PP is acceptable)
    if (qa < average_quality_value and not in the initial range)
        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)
        est = est1;
        dl = dl1;
        minq = average_quality_value * quality_reduce_ratio(0.6);
        muc = muc * cost_reduce_ratio(0.5);
    else
        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
    if (first counter proposal)
        if(dl1 - est1 < current_duration)
            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
            est = est1;
            dl = est + current_duration + delt_t;
            minq = average_quality_value * quality_increse_ratio(1.1);
    else
        if (qa > average_quality_value)
            est = st;
            dl = ft + delt_t;;
            minq = average_quality_value * quality_increase_ratio(1.1);
        else
            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
```

3.4 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 certain heuristics to control the search process. Earlier, we tried a binary search algorithm [11] whose short description follows. The contractor builds an initial proposal as described above: this initial proposal requests that the non-local task to be performed at the most convenient time for the contractor. If the contractee 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 contractor is a compromise of its own previous proposal and the contractee's counter proposal, while each next counter proposal from the contractee is the compromise of the contractor's proposal and the contractee's own previous counter proposal. Figure 2 and Figure 3 show how the contractor generates the new proposal based on its previous proposal and the counter proposal. The contractor 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 MUQ 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).

Let us see what actions the contractor takes if there is an existing acceptable commitment:

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

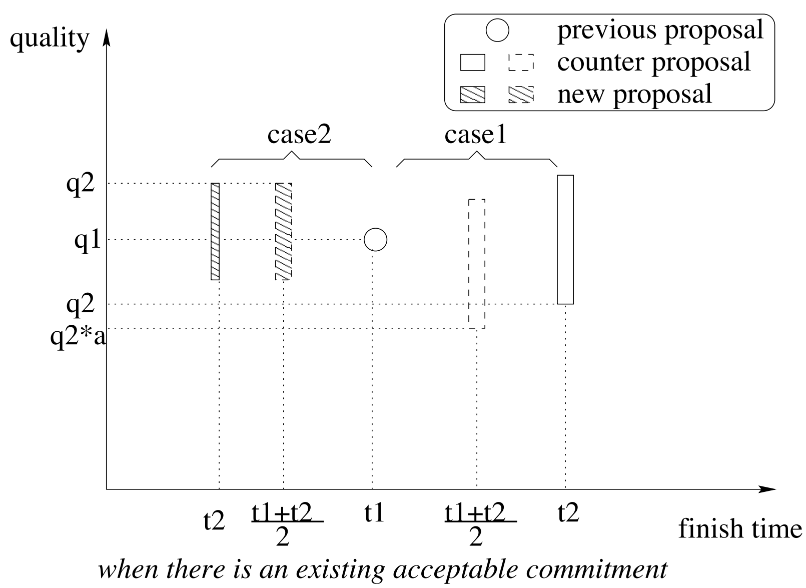

Figure 2: New proposal generation(with acceptable solution)

Let us see what actions the contractor takes if there is no existing acceptable commitment yet:

- The contractee informs the contractor that it cannot do task NL as early as the contractor requested, but it can do it later with a higher quality: the contractor now asks for a finishing time to be the average of those of the counter proposal and previous proposal and it keeps the requested quality the same, thus trying to meet the contractee halfway (Figure 3, case 2).
- The contractee informs the contractor that it cannot do task NL as early as the contractor requested and the quality requested is not possible: the contractor asks for a finishing time to be 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 3, case 3).
- The contractee informs the contractor that it can do task NL at the requested time or earlier and even with a higher quality than

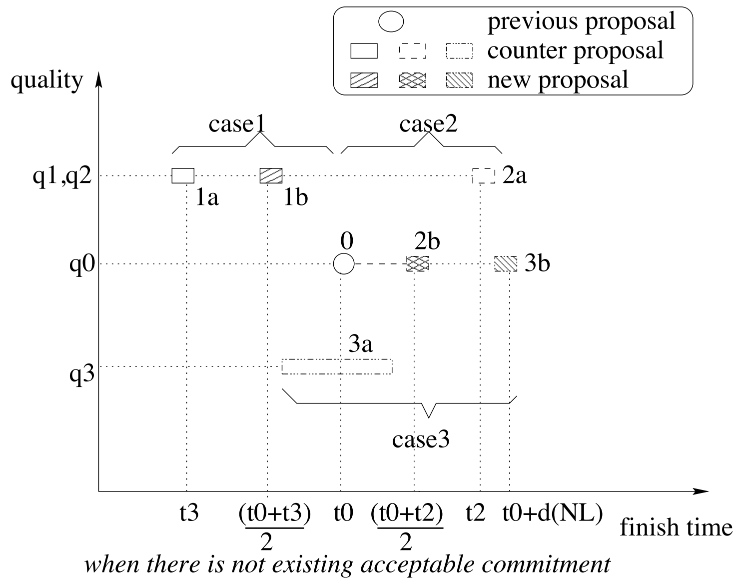

Figure 3: New proposal generation(no acceptable solution)

requested: the contractor asks for a finishing time to be the average of those of the counter proposal and previous proposal and request the quality as the contractee offered thus trying to see if this new pair reduces the contractee's cost (Figure 3, case 1).

- The contractee informs the contractor that it can do task NL at the requested time or earlier but the quality requested is not possible: the contractor asks for a finishing time to be 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 3, case 3).

This binary search algorithm does not work as well as the range-by-range search because it finds 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 help 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 contractee 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 some solutions often being missed, because both agents are likely to search only in the vicinity of their most favorite proposals.

3.5 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 work.

- SingleStep : The contractor sends a proposal commitment to the contractee, the contractee accepts PC if MUG(PC) $>$ MUC(PC); otherwise it rejects PC, and the negotiation is terminated in failure.
- MultiStep-Multiple(n)-Try : The contractor and the contractee 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. 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 contractor and the contractee perform the negotiation series - "proposal, counter proposal, new proposal, ..." - until certain iteration limits are reached. This protocol will explore 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 a 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 contractee 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 with their other local activities. Thus the computation effort for a complete search is not feasible. Hence, a range-by-range search (with step size of 5) is performed in the whole search space 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 contractor and the contractee. 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.

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

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

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[10].

3.6 Example

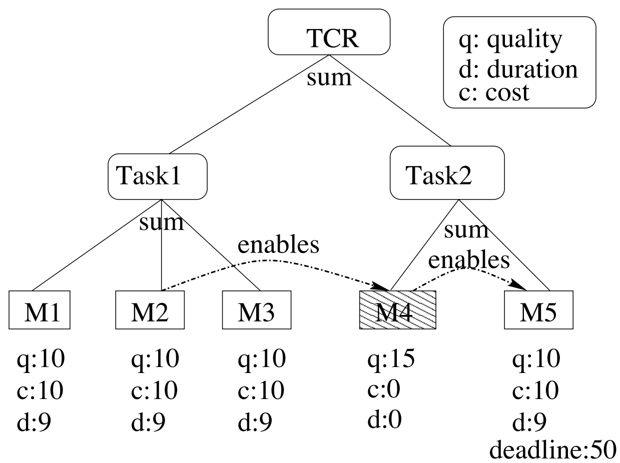

Figure 4: The contractor's task structure

In this section, we use an example to explain how the negotiation mechanism works.

For instance, the contractor is working on task TCR (Figure 4). TCR has two subtasks, Task1 and Task2. Task1 has three subtasks, M1, M2 and M3. Each of them takes 9 units processing time (d:9), has a cost of 10 (c:10) and generates 10 units 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). The contractee is an agent that could potentially perform task M4. (There could be more than one potential agent. For clarity we only show one). Similarly, Figure 4 shows the contractee's local task TCE (the left part of the figure).

In this example, the contractor 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 contractee 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 $duration\_weight = 0.1$.

Step1: Build-Proposal (Action A in Figure 1) The contractor schedules local task structure TCR assuming M4 is not to be done and gets the following schedule S1:

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

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

then it schedules TCR 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)

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

Then it builds the commitment PC0 based on S2: since Method2 enables Method4, so the earliest start time is 9; the deadline is 27 because it has to be finished before Method5's scheduled start time 27; the given range 18 here seems very flexible compared to the estimated duration(10.5), so the quality request is set to a higher value(18.0) than the average value(15.0) of the estimation quality achievement.

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

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

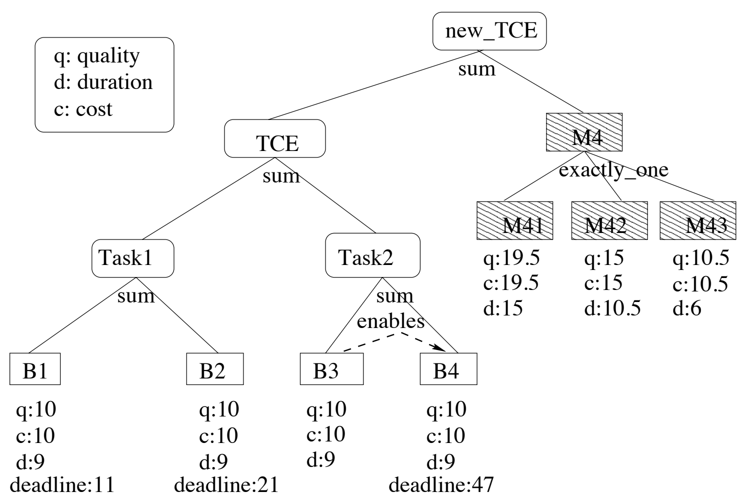

Figure 5: The contractee's task structure

Step2: Evaluate-Proposal (Action J in Figure 1) The contractee receives this commitment, adds M4 to its local task structure TCE and gets a new task structure new_TCE (Figure 5). The contractee 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 TÆMS structure.

The contractee schedules new_TCE 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(S3)=30[6]; Cost(S3)=49.5; Duration(S3)=42; Utility(S3)=0.446

Compared with the schedule S4 without performing Task M4:

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

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

[6]Notice the quality of schedule S3 does not include the quality achieved of M41 since it does not contribute to the contractee's local utility.

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

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

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

Step3: Re-Evaluate-Proposal (Action I in Figure 1) The contractor receives PC0 and re-evaluates it since it received a higher quality and the earlier finish time than it requested:

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

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

so this is an acceptable commitment. In either a SingleStep protocol or a MultiStep-One-Try protocol, the contractor 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 contractor continues negotiation and tries to find a better commitment.

Step4: Generate-New-Proposal (Action D in Figure 1) If the contractor decides to find another solution, it attempts to improve the proposal based on its previous proposal and the current proposal from the contractee. It constructs a new proposal by decreasing the quality request, as the algorithm described in Section 3.3:

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

The contractor 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)

$Utility(S5) = 0.83, MUG(PC1) = 0.274$

PC1 is sent to the contractee.

Step5: Evaluate-Proposal (Action J in Figure 1) The contractee 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.

Step6: Re-Evaluate-Proposal (Action I in Figure 1) The contractor receives PC1 and re-evaluates it based on the higher quality and the earlier than requested finish time it gets:

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

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

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

Step7: Generate-New-Proposal (Action D in Figure 1) It rebuilds a new proposal by moving the earliest start time later, from old start time 9 to 14 by adding 5 (the step size is 5), as the algorithm described in Section 3.3:

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

The contractor 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 contractee.

Step8: Evaluate-Proposal (Action J in Figure 1) The contractee 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.

Step9: Re-Evaluate-Proposal (Action I in Figure 1) The contractor 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_request: 10.5]

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

so this is a better acceptable commitment. In a MultiStep-Two-Try protocol, the contractor agent will stop and accept this commitment with the combined utility gain of 0.179. In a MultiStep-Three-Try protocol, the contractor continues negotiation and tries to find a better commitment.

Step10: Generate-New-Proposal (Action D in Figure 1) It rebuilds a new proposal by requesting a higher quality and extending the deadline, as the algorithm described in Section 3.3:

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

The contractor 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 contractee.

Step11: Evaluate-Proposal (Action J in Figure 1) The contractee 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.

Step12: Re-Evaluate-Proposal (Action I in Figure 1) The contractor receives PC3 and re-evaluates it based on the higher quality and the earlier finish time it gets than requested:

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

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

so PC3 with the combined utility gain of 0.214 is the best solution found so far. In a MultiStep-Three-Try protocol, the contractor 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.

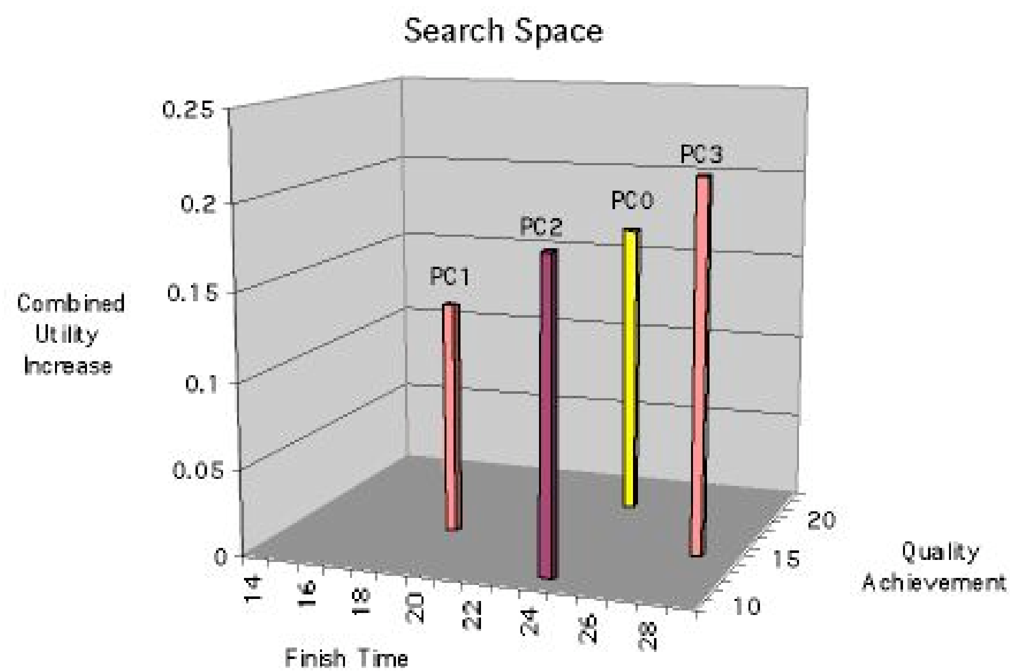

Figure 6: Search space

By now, the contractor has obtained four acceptable commitments: PC0 starts from 9 and finishes at 24, achieves quality 19.5 and has a combined utility increment of 0.169, PC1 starts from 9 and finishes at 19, achieves quality 15 and has a combined utility increment of 0.133, PC2 starts from 18 and finishes at 24, achieves quality 10.5 and has a combined utility increment of 0.179, PC3 starts from 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 experiment is designed to examine how different protocols work in different situations and find what major factors affect the negotiation outcomes. Two agents have been constructed, the contractor agent and the contractee 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 4 and Figure 5. 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 4, 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 difficulties. There is a total of 4096 ($2^6 * 2^6$) test cases obtained from the combinations of these task structures. Figure 7 shows the contractor's task structure and the contractee's task structure with all six possible temporal constraints and all six possible interrelationships. Besides the five different protocols described in section 3.5, we also developed a "complete search" algorithm as a comparison base for the experiment. The "complete 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 "complete 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 solution". We compare the solution from each protocol to the best solution to evaluate its effectiveness.

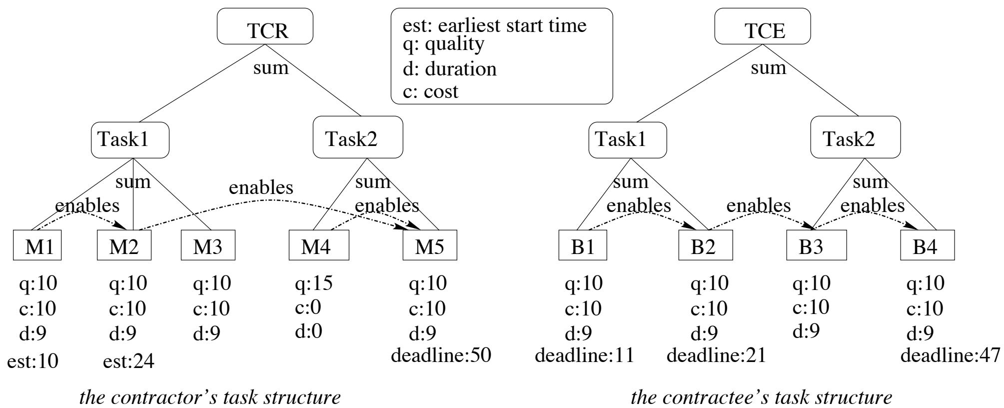

Figure 7: Examples of various task structures

We collect the following data for each test case in the experiment:

- *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 out of the combined utility without performing the task allocation.
- *Solution Quality*: How good this solution is compared to the best solution from the "complete search". We compare only the utility increase from the negotiation. Suppose a negotiation solution results in the combined utility increased by 18%, and the best negotiation solution could 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*: The number of constraints ("deadline" and "enables" relationships) in the task structures that are mapped onto the complexity measure of the negotiation. The formula we use to calculate the complexity 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 contractor's task structure; tc1: number of temporal constraints in the contractor's task structure; ir2: number of interrelationships in the contractee's task structure; tc2: number of temporal constraints in the contractee'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 would be.

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

	Success	AGP	ANNS	GPS	SQ
SingleStep	2850	7.63	1.0	7.63	51.44
MultiStep-One-Try	4088	10.17	1.48	6.87	72.37
MultiStep-Two-Try	4088	11.9	4.69	2.55	84.57
MultiStep-Three-Try	4088	13.4	6.42	2.09	96.21
MultiStep-Limited-Effort	4088	13.9	8.15	1.7	99.36

Table 1: comparison of protocols (AGP: the average of the gain percentage over all cases. ANNS: the average number of the negotiation steps over all the cases. GPS: the negotiation gain over each step (GPS=AGP/ANNS). SQ: the average of the solution quality over all cases.)

Table 1 shows the comparison of these five protocols. Out of the 4096 test cases, the SingleStep protocol succeeds in 2850 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 solution, the average number of the negotiation steps(ANNS) is 1, 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 the negotiation gain over each step (GPS) decreases from 6.87% to 1.7%.

Figure 8 shows how these five protocols perform when the complexity of the task structures changes. 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), its performance decreases dramatically 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 have a solution quality that is only 10% of the best 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 solution; with MultiStep-Three-Try, the lower bound of the solution quality is raised to 70% of the best solution. The agent can

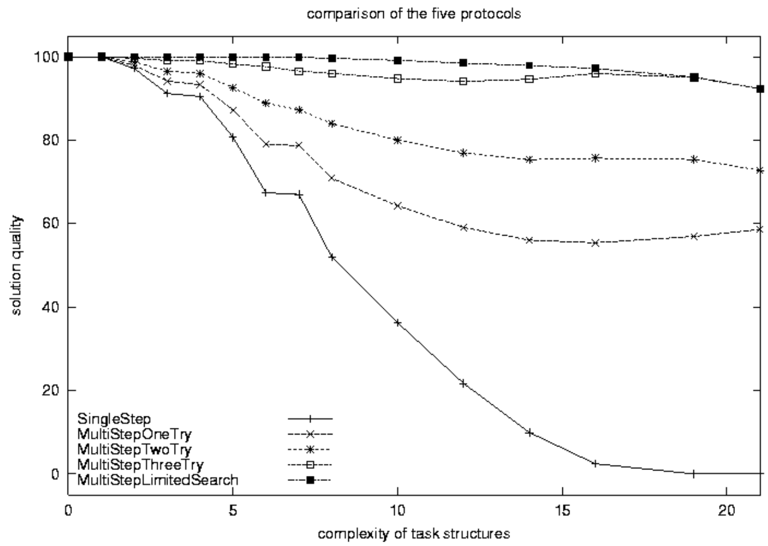

Figure 8: Comparison of five protocols according to complexity (the solution quality is a relative quality compared to the best solution, number 100 means the best solution)

choose the 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 probably could reduce the utility. For example, the contracted task 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 available for negotiation without affecting the earliest start time of the task; and the frequency of new tasks arriving, opportunity cost and so forth. Giving 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(n) of Negotiation Steps* can be mapped into certain Utility Percentage (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 the figure 10 is calculated as the following formula:

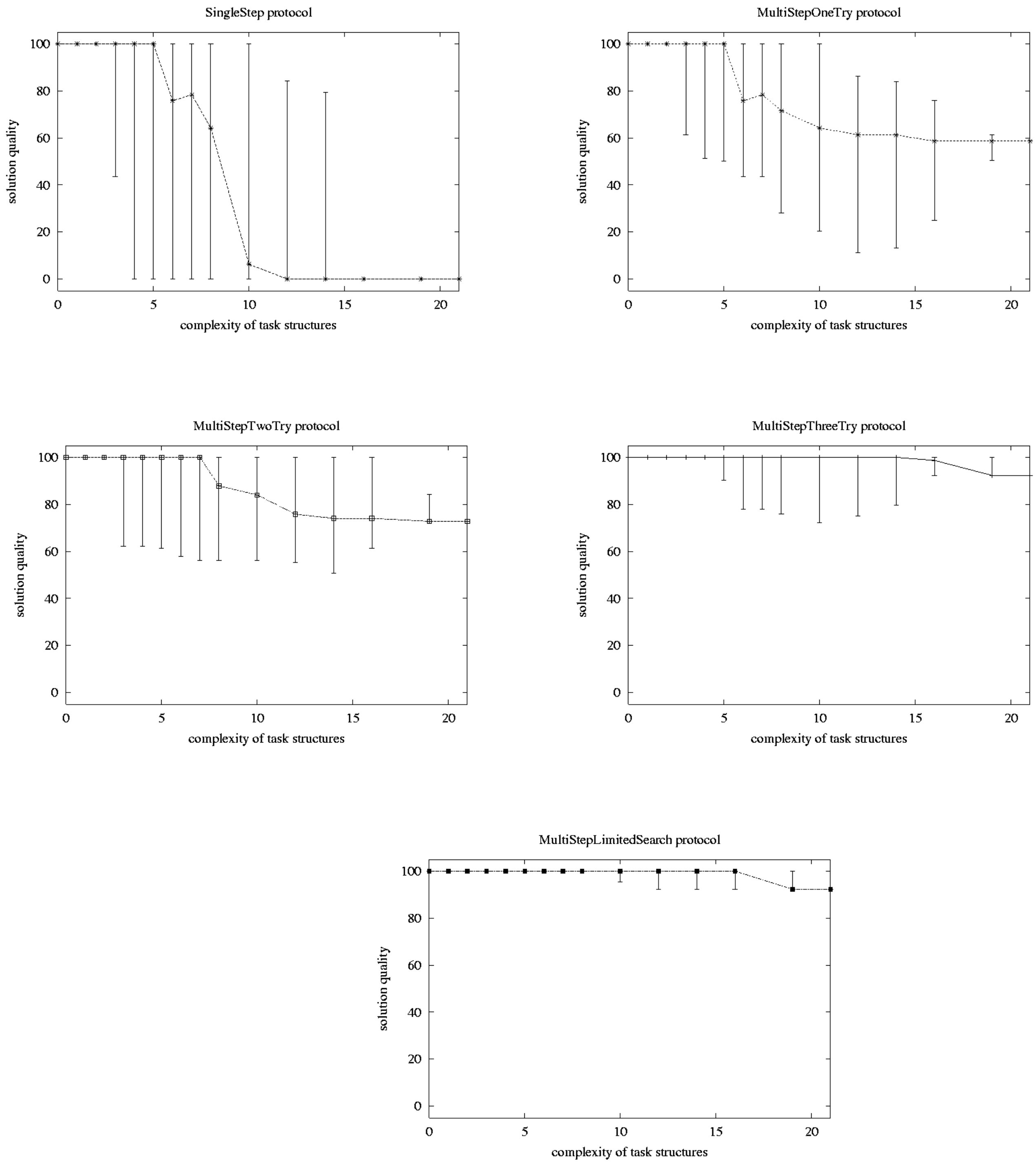

Figure 9: The performance of all protocols with error bars

$$net\_negotiation\_gain = \frac{\frac{combined\_utility\_after\_negotiation - initial\_combined\_utility\_without\_negotiation}{initial\_combined\_utility\_without\_negotiation} - c*n}{\frac{best\_solution\_gain}{initial\_combined\_utility\_without\_negotiation}} * 100$$

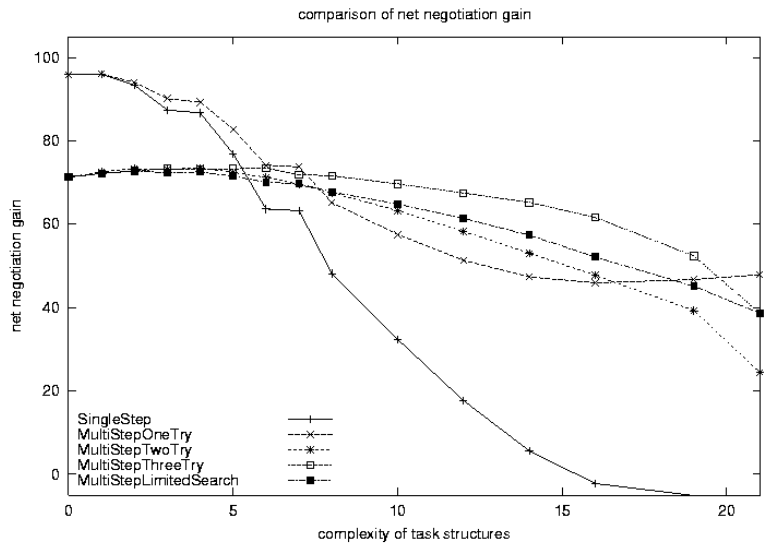

Figure 10: Negotiation gain beyond effort

Figure 10 shows the comparison of the *net negotiation gain* (the negotiation gain minus the negotiation effort) of the five protocols. There is a phase transition phenomenon: 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 here. 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 all kinds of situations 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. The contractee agent can inform the contractor agent of its local constraints number before the negotiation process starts, and the contract agent can decide which protocol to choose (how much effort to put in the negotiation) according to the estimate of the negotiation difficulty.

Figure 11 shows each protocol's performance under the three different situations: low complexity, medium complexity and high complexity. The table shows how much the negotiation gain (AGP) is and how much the negotiation cost (ANNS) is. 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

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

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 idea in another way.

Based on the above mentioned 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 much more gain with less 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 Number of Constraints can be used to choose the appropriate protocol that balances the negotiation gain and negotiation 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 need to 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 it can't capture more detailed information. Hence, we have done additional work trying to find a better predictor.

5.1 Solution Space Analysis

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 complete search was performed and all solutions were

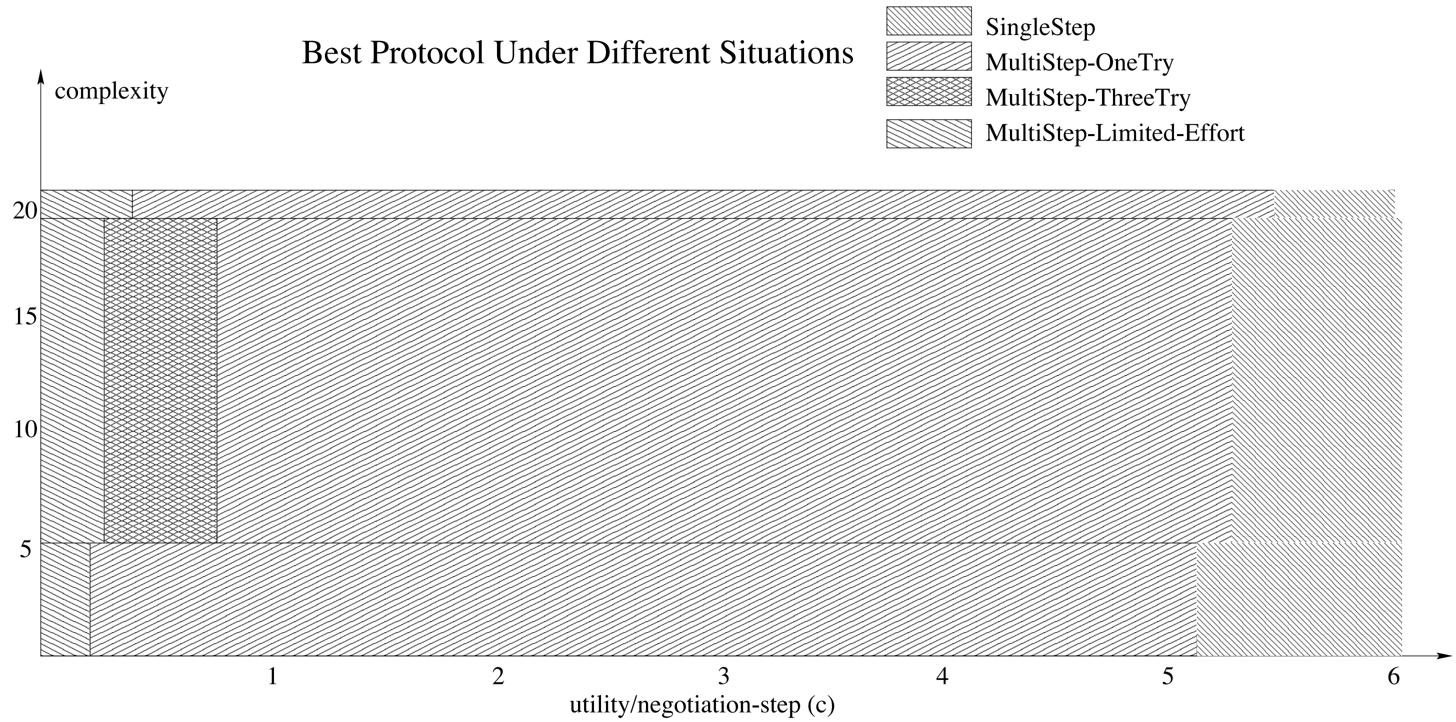

Figure 12: Best protocol under different situations

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 all solutions;
2. Unique Solution Number (usn): the number of unique solutions: does not count different approaches at the same start time;
3. Good Solution Number (gsn): the number of good solutions: if the solution is better than 3/4 of the best 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 is 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.

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 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.

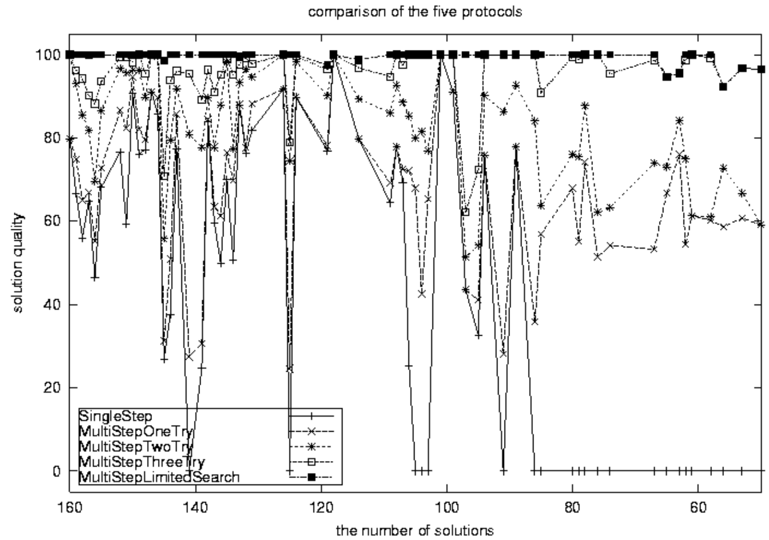

Figure 13: Comparison based on solution number

From the above observation we find that it is not only the solution number 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 [2]) that measures how flexible a set of tasks is:

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

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 p(i, j) = 1, then task T_i must be finished before task T_j ; otherwise they can be performed in any order.

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

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

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 the flexibility measure is: the more slack time the tasks have, and the fewer precedence relationships among those tasks, the greater the flexibility is. If the agent has more slack time for its local tasks and fewer precedence constraints among them, it is easier for it 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 contractor has a task structure (TSa) and the contractee has a task structure (TSb), and the contractor needs to assign the nonlocal task Tnl to the contractee.

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))

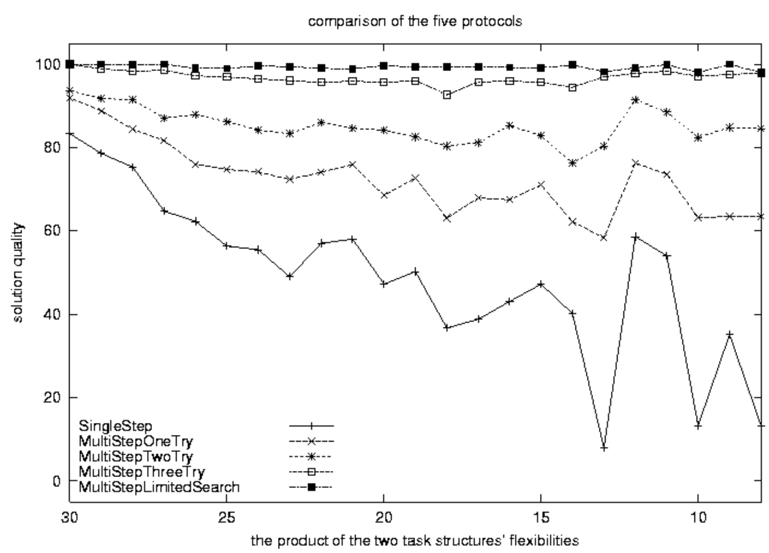

Figure 14: Comparison based on fproduct

Almost all these measures (except the fmin measure, seems it is not a good measure for the combination of two flexibility measures) provide results similar to the complexity measure: as the flexibility related measure decreases, the difference among the performance of those protocols increases. Figure 14 and 15 show the comparison of the five protocols according to the fproduct and the tfproduct measures. Why couldn't the flexibility related measures work better? One reason is that the flexibility measure only deals with the slack time that is calculated from the hard deadlines; thus it can't reflect the soft deadline idea [7].

[7] The hard deadline means that the task produces zero utility if it could be be finished before the deadline; while the soft deadline allows a task generate certain quality even it is finished after the deadline. However the earlier the tasks are finished, the higher the utility could be (see definition of utility function in section 3.5)

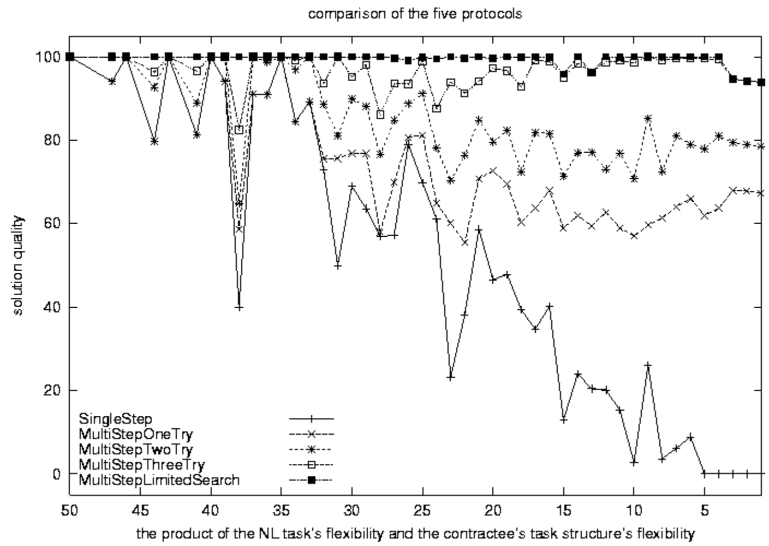

Figure 15: Comparison based on tfproduct

5.4 Initial Range Related Measure

We also found the initial proposed range for the nonlocal task is an important factor that affects the protocols' performance, following two measures based on the initial range:

1. Initial Range and Constraints Number Measure (rconst): rconst = (8- Initial_Range(Tnl)/5) + ir2+tc2 [8]

2. Initial Range and Flexibility Measure (rflex): rflex = Initial_Range(Tnl)/5 + F(TSb)*10

These two measures also provides results similar to the complexity measure as shown in Figure 16 and 17.

We have tried all 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 unexpected.

6 Conclusions & Future Work

In this paper, we present a cooperative negotiation mechanism in which negotiating occurs over a multi-dimensional utility function. We show the application of this mechanism to the task allocation domain in a cooperative system. The contractor agent has a task that need to be performed by other agents. To perform this task, the contractee agent could 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 started with a binary search algorithm as a mechanism to find a compromise between the contractor's protocol and the contractee's counter proposal. After examining the trace of this negotiation mechanism

[8] ir2: number of interrelationship in TSb ; tc2: number of temporal constraints in TSb.

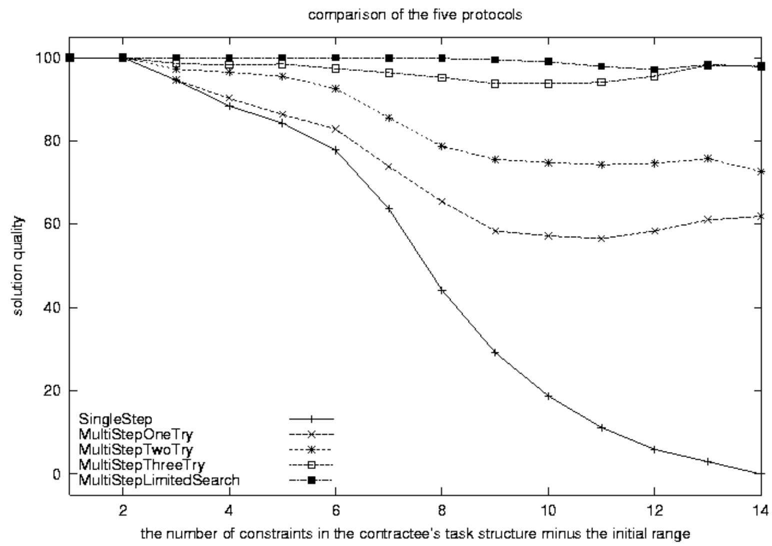

Figure 16: Comparison based on rconst

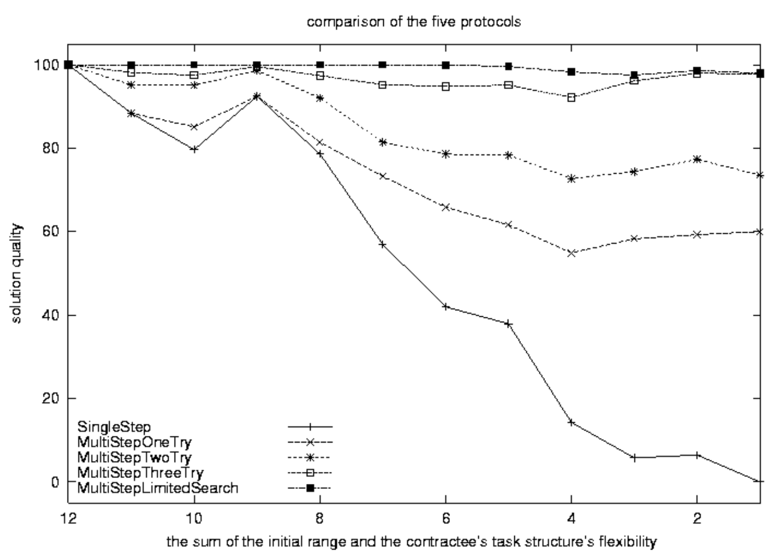

Figure 17: Comparison based on rflex

carefully, we developed a better way to do the distributed search explicitly in the agents' negotiation. The range-by-range algorithm searches a boarder 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 contractee agent to have more freedom to react, which improves the efficiency of the negotiation. The MultiStep negotiation mechanism is actually an anytime mechanism: by investing more time, the agent may find a better solution. A set of MultiStep protocols are developed based on this mechanism. Experimental work is done to study how different protocols work in different situations. A complete search is performed as a baseline for comparison. We find a phase transition phenomenon: when the negotiation situation is very simple or very difficult, the extra negotiation effort does not bring reasonable extra gain. When the negotiation situation is of medium difficulty, the extra gain exceeds the extra effort. The 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 how difficult the negotiation could be to help the agent to 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 will continue our work in two directions. One is to extend this mechanism to multi-linked negotiation where multiple related negotiation issues occur simultaneously. Another one is to obtain a better understanding of the negotiation problem characteristics. These characteristics should help us rate negotiation problems by their difficulty, estimate the probability of finding a good solution before the negotiation is even started, and ultimately, 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 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] Decker, K., Lesser, V. R. Quantitative Modeling of Complex Environments. In International Journal of Intelligent Systems in Accounting, Finance and Management. Special Issue on Mathematical and Computational Models and Characteristics of Agent Behavior, Volume 2, pp. 215-234, 1993.

[2] Deshmukh, A. V., Talavage, J. J., and Barash, M. M. Complexity in Manufacturing Systems: Part 1 - Analysis of Static Complexity IIE Transactions, vol 30, number 7, pp. 645-655, 1998.

[3] Lander, S. and Lesser, V. Understanding the Role of Negotiation in Distributed Search Among Heterogeneous Agents. In Proceedings of the International Joint Conference on Artificial Intelligence, 1993.

[4] Laasri, B., Laasri, H., Lander, S. and Lesser, V. A Generic Model for Intelligent Negotiating Agents. In International Journal on Intelligent Cooperative Information Systems, Volume 1, Number 2, pp. 291-317, 1992.

[5] O'Hare and N.R.Jennings (eds), 1996. Negotiation Principles. Chapter 7. Foundations of Distributed Artificial Intelligence:John Wiley & Sons, Inc.

[6] Moehlman, T., Lesser, V., and Buteau, B. Decentralized Negotiation: An Approach to the Distributed Planning Problem. In Group Decision and Negotiation, Volume 1, Number 2, pp. 161-192, 1992, K. Sycara (ed.), Norwell, MA: Kluwer Academic Publishers.

[7] Sandholm, T. 1993. An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations. Eleventh National Conference on Artificial Intelligence (AAAI-93), Washington DC, pp. 256-262.

[8] Gerhard Weiss (ed). 1999. Chapter 5, Distributed Rational Decision Making. Multiagent System:MIT press

[9] Thomas Wagner, Alan Garvey, and Victor Lesser. Criteria-Directed Heuristic Task Scheduling. Intl. Journal of Approximate Reasoning, Special Issue on Scheduling, 19(1-2):91–118, 1998. A version also available as UMASS CS TR-97-59.

[10] Wagner, Thomas A., Garvey, Alan J. and Lesser, Victor R. Complex Goal Criteria and its Application in Design-to-Criteria Scheduling. In Proceedings of AAAI-97, pp. 294-301, Providence, Rhode Island, August 1997. (A version is also available as UMASS Computer Science Technical Report 1997-10.)

[11] Zhang, XiaoQin; Podorozhny, Rodion; Lesser, Victor. Cooperative, MultiStep Negotiation Over a Multi-Dimensional Utility Function. In Proceedings of the IASTED International Conference, Artificial Intelligence and Soft Computing (ASC 2000), 136-142, Banff, Canada, July, 2000, IASTED/ACTA Press.