THE ROLE OF COMMITMENT IN COOPERATIVE NEGOTIATION

Sandip Sen
Mathematical & Computer Science Dept
University of Tulsa
600 South College Avenue
Tulsa, OK 74104, USA
sandip@kolkata.mcs.utulsa.edu

Edmund H. Durfee
EECS Dept
University of Michigan
1101 Beal Avenue
Ann Arbor, MI 48109, USA
durfee@engin.umich.edu

ABSTRACT
Cooperative information agents need mechanisms that enable them to work together effectively while solving common problems. We investigate the use of commitment by agents to proposed actions as a mechanism that allow agents to work concurrently on interdependent problems. Judicious use of commitment can not only increase the throughput of cooperative information systems, but also allow them to deal flexibly with dynamically changing environments. We use the domain of distributed scheduling to demonstrate that static commitment strategies are ineffective. Results from simulated experiments are used to identify the environmental features on which an adaptive commitment strategy should be predicated.

Keywords: information agents, distributed scheduling, adaptive commitment

1. Introduction

We are interested in designing intelligent software agents that can be used to automate routine cooperative information exchange and problem-solving in human organizations. As a part of that objective, we have been developing intelligent, autonomous, adaptive agents for distributed scheduling problems. The scheduling agents negotiate by exchanging information about resources that each is managing in order to collectively accomplish desired tasks. In order to perform their assigned tasks competently, each of these agents must be able to use effective problem-solving strategies for the target domain, be able to adapt its problem-solving behavior to suit changing demands on the system as specified by dynamic environmental conditions, and be able to explain its actions to end users as and when requested.

To set the broader context for our work, we now define what we mean by the term cooperative information systems:

Definition: *A collection of information agents with varying functionality and composition, but with shared goals that necessitate the cooperative exchange of restricted and pertinent information, constitute a cooperative information system.*

In our application domain, the individual information agents are intelligent task schedulers that manage task scheduling requests on behalf of associated humans. To successfully schedule a task that requires one or more resources, the scheduling agents corresponding to each of the required resources must cooperatively exchange resource availabilities. Although exchanging entire calendars allows for scheduling of a task in one iteration, it puts unnecessary load on the communication channel. Additionally, this will lead to redundant information processing and serialization of task requests that could have been scheduled concurrently. Hence, we need intelligent negotiation mechanisms to enable scheduling agents exchange only relevant information. The use of a well-understood problem solving protocol allows agents with heterogeneous representation for local information to negotiate intelligibly with one another, and consequently, increase the interoperability of the system.

In our work, we have used an easily understood model of agent design and interaction: agents are designed to establish and carry out contracts, which commit resources to tasks at particular times. Yet, a wide variety of results can emerge within the contracting model, depending on the agents' decisions about how to rank possible commitments, about how to treat tentative commitments (proposed but not yet confirmed task schedules), and about when to withdraw from previous commitments (canceling and rescheduling tasks). Performance of an agent will be dependent on the options available for each of the decisions it has to make. This necessitates a thorough investigation of different heuristic strategies for commitment that an automated agent can use.

It is necessary that we clarify what we mean by the term *commitment*. We assume the following definition of the term [6]:

Definition: *Commitment is an agreement or pledge to do something in the future.*

An agent can make such a pledge to itself, or to another agent. In this paper, we will concern ourselves with the decision regarding whether or not to commit to proposed times to schedule an activity. For an example, consider the case of an agent who has agreed to provide a resource to another agent who requested it for an activity on Monday morning. Before receiving a confirmation from this agent that the activity is indeed scheduled for the requested time, a third agent requests the resource for the entire day on Monday. Should the agent with the resource commit to its previous proposal and decline the request of the third agent, or should it not commit to the previous proposal and agree to supply the resource for the requested time to the third agent? The former would be counter-productive if the agent who requested the resource for Monday morning fails to schedule the corresponding activity at that time, but, in the latter, if both the requesting agents decide to schedule their respective activities at the proposed time, there will be a conflict for the non-sharable resource. In this paper, resources are definitely committed to activities once those activities are actually scheduled and all concerned agents are notified. For a discussion of cancellation and rescheduling, we refer the readers to our previous work [11].

Commitment will have an effect only on interdependent activities, and does not have any effects on independent activities (e.g., if we have one conference room, and 3 meetings to schedule, two of them being group meetings that need to use the conference room, whereas the third one can take place in the office of an individual, scheduling one group meeting in the conference room constrains the scheduling of the other group meeting, but does not affect the third meeting, assuming that none of the attendees to the third meeting is an attendee to the scheduled group meeting). As such, commitment becomes necessary when interdependent activities are being scheduled concurrently (if only one activity is scheduled at a time, whether agents commit to proposed schedules or not is irrelevant). Commitment strategies become particularly relevant in dynamic domains, where all the problems to be solved are not known ahead of time, and in the midst of negotiation over one problem, a new, interdependent problem may arrive. If used appropriately, commitment can aid multiagent negotiation to quick convergence. Using commitment injudiciously, however, can severely affect the performance of cooperative agents. This paper is an investigation into the factors and conditions to be considered by cooperative agents to determine an effective commitment strategy.

In this paper, we first motivate the need for commitment in multiagent negotiation. Next, we analyze the possible effects of two simple commitment strategies on concurrent scheduling processes. Our analysis shows that each of these commitment strategies produce some beneficial and some harmful interactions between scheduling processes. In order to evaluate the relative frequencies of the harmful and beneficial interactions for the commitment strategies, we ran simulation experiments on a distributed scheduling problem. Results from these experiments suggest that a decision to always commit during negotiations can severely affect the quality of problem solving. This leads us to the identification of the environmental conditions on which an adaptive commitment strategy, that chooses the commitment option to allow scheduling of more tasks of higher priority given the current calendar and task requests, should be predicated. We conclude the paper by outlining the benefits of an adaptive commitment strategy in negotiation between cooperative agents.

2. The need for commitment

In this section, we motivate the need for commitment in multiagent negotiation. We also compare and contrast our work on commitment with that of other researchers interested in studying the role of commitment in multiagent problem solving.

In order to achieve local and global goals in a multiagent environment, an intelligent agent needs to model the expected future actions of other agents. In cooperative settings, communication provide a means for collecting information to help this modeling process. But information being communicated is of little help in deciding future actions if it is not persistent. The use of commitment by an agent allows other agents to model some of its future course of actions accurately, and hence can lead to more coordinated problem solving. But commitment to a course of action by an agent does limit its flexibility to respond to new contingencies (assuming that commitments cannot be violated). Although the above analysis seems to suggest that commitment is useful to the recipients and not to the agent making the commitment, an agent can take a lead in a decision-making situation by making a commitment to its preferred activity, and thus persuade others to adopt a course of action deemed suitable to the needs of the committing agent. In fact, it has been shown that commitment can be used even by non-cooperative (antagonistic) agents to gain advantage in hostile negotiations [3, 9]. In cooperative settings, however, reciprocal commitments enable quicker convergence in multiagent negotiations when compared to unilateral commitments.

Cohen and Levesque [1] have developed the semantics associated with committing to a particular set of actions. Jennings [4], on the other hand, emphasizes the centrality of commitment in coordinating multiple agents. The latter work also concerns the use of conventions as a guide to reasoning when agents fail to honor their commitments. Although these researchers stress the role of commitment in coordinating multiple agents, they do not address the critical questions of what to commit to and when to commit to it. In this paper, we address these critical questions as pertinent to the domain of distributed task scheduling. Thus, we view our work as complementary to other work done in the field on the use of commitment to aid coordination.

In distributed scheduling, in the period between the time an agent proposes the availability of a resource for a task over some time interval, and the time when the agent hears back from the agent requesting the resource (either accepting or rejecting the proposal), the proposing agent has a choice to use or not use the proposed time interval to negotiate over another task. The rationale behind a good commitment strategy would be that commitment to the proposed interval is preferable if the agent receiving the proposal is likely to accept it, otherwise commitment can be wasteful (can prevent scheduling of other tasks that could have used the resource for the proposed interval). In the rest of this paper, we develop more precise metrics to aid a cooperative agent make informed commitment decisions.

3. A Formal Definition of Distributed Task Scheduling

A task schedule consists of a group of tasks to be scheduled using a set of resources managed by a set of agents. Given a set of n tasks and k resources, a scheduling problem is represented as $\mathcal{S} = (\mathcal{A}, \mathcal{T})$, where $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ is the set of resource managers (one manager per resource) and $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ is the set of tasks to be scheduled. A time *slot* is represented as a date, hour pair $\langle D, H \rangle$. A set of contiguous time *slots* is called a time *interval*. A task named i is represented by a tuple:

$$\tau_i = (A_i, h_i, l_i, w_i, S_i, a_i, d_i, T_i),$$

where

$A_i \subseteq \mathcal{A}$, is a set of managers of required resources for the task;
$h_i \in A_i$, is the host agent who coordinates the scheduling of the task;
l_i is the required duration of the task in hours;
w_i is the weight or priority assigned to the task;
S_i contains a set of possible starting times on the calendar for the task.
$a_i = \langle D_{a_i}, H_{a_i} \rangle$, is the time at which h_i becomes aware of the need to schedule τ_i;
$d_i = \langle D_{d_i}, H_{d_i} \rangle$, is the deadline by which the host h_i needs to schedule the task τ_i;

T_i is the time interval for which the task τ_i is finally scheduled and is represented by an ordered set $\{\langle D_i, H_i \rangle, \langle D_i, H_i + 1 \rangle, \ldots, \langle D_i, H_i + l_i - 1 \rangle\}$, (here D_i gives the date and H_i gives the starting hour for which task τ_i is scheduled) if the task could be scheduled, and by $\emptyset$ otherwise.

We assume that resources cannot be shared by tasks, but this assumption can be removed with minor modifications to our protocol. Each resource is managed by a corresponding agent, who negotiates with other agents to schedule tasks that require that resource. A simple contracting protocol is used for negotiation. The host agent announces the task to each of the agents managing the required resources for the task. The host also suggests one or more suitable intervals (intervals at which it can use the local resource for the task) to the other resource agents. In our implementation, the host proposes 3 intervals per iteration of negotiation from the least dense part of the schedule of its associated resource. Each resource agent submits bids (counter-proposals) in response to the proposals received, containing intervals at which they can provide their resources for the task. If a common time interval is found at which all resources are available, the task is scheduled. Otherwise, the host agent asks resource agents to provide their resources at some other preferred time intervals. This cycle continues until an acceptable time interval is found, or it is recognized that the task being negotiated cannot be scheduled. We assume a dynamic environment where tasks arrive over time. This problem is more difficult than the static scheduling problem where all tasks are available before processing starts [14]. Furthermore, we require the same task be processed using multiple resources to be acquired from different sources. This is considerably more problematic than the situation where each task can be processed by using just one resource.

4. Scheduling processes

Our scheduling agents handle the demand of concurrent scheduling of multiple tasks by creating a process for each such task. The different concurrently active processes, forked by managers of different resources required by a task, negotiate using the multistage negotiation protocol [2], an extension of the contract-net protocol [13]. Each scheduling agent manages the calendar accesses by the different processes it has forked. We do not concern ourselves with concurrency control mechanisms, assuming the availability of standard database or file access control mechanisms [5, 8].

A given task τ_i is scheduled through negotiation by $|A_i|$ processes, one for each of the resources required for the task. The process created by the agent managing the jth resource to schedule task i is referred to as $\mathcal{T}_{ij}$. A process $\mathcal{T}_{ij}$ will interact with other processes in two possible ways. First, because agents managing resources required by a task must exchange information to converge on a schedule, $\mathcal{T}_{ij}$ will interact via communication with other $\mathcal{T}_{ik}$ processes — the processes forked by managers of other resources required by task τ_i. Second, because the same resource manager has separate processes for scheduling the different tasks that need the corresponding resource, $\mathcal{T}_{ij}$ will interact with $\mathcal{T}_{lj}$ for other tasks τ_l that are being concurrently negotiated by agent j—this interaction takes place in the processes' contention for the shared calendar for the corresponding resource.

4.1. Resource requirements of scheduling processes

In this paper, we consider two simple commitment strategies: **committed** or **non-committed**. The choice of committing or not committing to a proposed time interval amounts to either blocking or not blocking valuable calendar resources until complete agreement is reached. Commitment can cause non-optimal schedules as some tasks block time intervals that cause other tasks to be abandoned due to lack of uncommitted times within the task's constraints. In some instances, those blocked intervals might later be released. On the other hand, blocked time intervals prevent attempts to propose overlapping time intervals for two different tasks, which can save scheduling time and the amount of information exchanged to schedule tasks. So, although the primary effect of commitment is on the percentage of requested tasks that can be scheduled, this strategy choice also affects the total time taken and proposals exchanged in the scheduling process.

Viewing a commitment strategy as affecting the interaction between processes that require common resources, we can formally represent the resource requirements of process $\mathcal{T}_{ij}$ as a 4-tuple,

$$\Re(i,j) \;=\; (v_{ij}, p_{ij}, b_{ij}, r_{ij})$$

where

- v_{ij} represents the set of all *viable* time intervals that could have been proposed for task τ_i by agent j. It is given by the set of all time intervals of length l_i whose starting slot belongs to S_i (S_i is the set of possible starting times on the calendar for task i).

- p_{ij} represents the set of time intervals that have been *proposed* by agent j and are still being considered for task τ_i. $p_{ij} \subseteq v_{ij}$, since only viable time intervals are proposed.

- b_{ij} represents the set of time intervals that have been *blocked* for probable use by agent j for task τ_i. These time intervals are under active consideration, but at most one of these will be used for the task. $b_{ij} \subseteq p_{ij}$, since only a subset (possibly empty) of the proposed time intervals can be blocked.

- $r_{ij} = T_i$ if τ_i has been scheduled (represents the time *reserved* for the task), and is $\emptyset$ otherwise. $r_{ij} \subseteq p_{ij}$, since the finally reserved time interval for a task is one on which all attendees have agreed and hence must have been proposed.

For process $\mathcal{T}_{ij}$, v_{ij} represents the static part of resource requirement, p_{ij} and b_{ij} are the dynamic parts, and, assuming no cancellation, r_{ij} changes at most once (from $\emptyset$ to $T_i \neq \emptyset$ if the task could be scheduled) during the lifetime of the process.

4.2. Types of interaction through shared resources

In this section, we identify different modes of interaction between two processes that are sharing the same resource (a resource's calendar) to schedule different tasks. Scheduling inefficiency arises due to interaction via shared resources, when two such processes try to use overlapping time intervals to schedule their respective tasks. Commitment strategies determine a large percentage of these interactions.

Given the definitions in Section 4.1. for viable, proposed, blocked, and reserved time intervals, we predict that the following kinds of interactions can take place between two processes $\mathcal{T}_{x,j}$, $\mathcal{T}_{y,j}$, $x \neq y$ both of which require the resource j:

possible: $\exists X, Y,\ X \in p_{xj}, (\exists y, Y \in p_{yj}), X \bigcap Y \neq \emptyset$. If overlapping time intervals have been proposed for different tasks by the respective processes, there is a possibility that both these tasks could be scheduled for these time intervals, in which case one of the processes has to retract and try to schedule the corresponding task at some other time interval.

actual: $\forall X, \exists Y,\ X \in v_{xj}(\exists y, Y \in r_{yj}), X \bigcap Y \neq \emptyset$. This scenario corresponds to the case where a request for a task τ_x comes in such that all viable time intervals corresponding to that task overlap with reserved time intervals for some other task (τ_y). In such a case, the processes $\mathcal{T}_{xj}$ and $\mathcal{T}_{yj}$ are actually competing for overlapping intervals of time and this results in a failure to schedule task τ_x.

preemptive: $\exists X, Y, Z,\ X \in v_{xj}, (\exists y, Y \in b_{yj}, X \bigcap Y \neq \emptyset) \bigwedge \neg(\exists z, Z \in r_{zj}, X \bigcap Z \neq \emptyset)$. This scenario corresponds to the case where a request for a task τ_x comes in such that at least one viable time interval corresponding to that task overlaps with a blocked time interval for some other task (τ_y), but does not overlap with any reserved time intervals. If $\forall X, \exists Y, Z, X \in v_{xj}(\exists y, Y \in b_{yj}, X \bigcap Y \neq \emptyset) \bigwedge \neg(\exists z, Z \in r_{zj}, X \bigcap Z \neq \emptyset)$, it is not possible to schedule τ_x.[1]

From the above interactions, **actual** interactions can happen with both the commitment strategies, **possible** interactions can happen only with the **non-committed** commitment strategy, and the **preemptive** interactions can happen during any iteration with the **committed** commitment strategy and only the final confirmation iteration with the **non-committed** commitment strategy. The above analysis concentrates on the effect of commitment strategies on the success rate of scheduling tasks. These strategies also impact the number of rounds of negotiation required to schedule a task. The above analysis also does not provide any estimate about the relative frequency of these different interactions. The following experiments were conducted to obtain that information.

4.3. Experiments

In order to better understand the effect of commitment on the success likelihood of scheduling tasks and the effort required to schedule them, we performed experiments where negotiation density (number of tasks being negotiated concurrently) and schedule densities (fraction of resource schedules used by tasks) were varied as independent parameters. The code for these experiments are written in the C

[1]Note that the scheduling process does not wait to see if the blocked time interval is actually used or not, but simply signals a failure to schedule the new task. This design decision was incorporated to prevent deadlocks. The process could time out after some pre-specified time period, but then the scheduling process will slow down considerably.

language, and the experiments were performed on a Silicon Graphics Indigo 3000 machine.

The experimental setup is as follows – a number of agents, starting with empty calendars for their associated resources, are given a set of tasks to schedule. Commitment strategies are held constant at either **committed** or **non-committed**. We allow simultaneous negotiations over different tasks; different hosts are negotiating on different tasks concurrently. We vary the arrival rate of task requests to the system, ranging from some minimum to maximum number of task requests being introduced into the system per time step of the discrete event simulation. An additional restriction is imposed on the agents: an agent can work on scheduling only one task at each time step. This constraint is implemented as follows: an agent looks at its input queues, and works on the earliest task request that has arrived and has not yet been processed. This may be a request to host a new task, proposing a time interval for a task for which bids have been received from all the invitees, or a counter-proposal to a proposal from another host. The other parameter that is varied is the desired density of the calendars (the final density of scheduled tasks on resource calendars assuming that each requested task was successfully scheduled). The effects of commitment on the iterations required to schedule a task, and on the task hours missed (number of task hours per agent that could not be scheduled) metrics are plotted against the rate of task arrivals and the desired schedule density (Figures 1 and 2). The corresponding plots for the case when proposed time intervals were not committed can be found in Figures 3 and 4.

The number of agents (and hence resources) in the simulations is 10, each working with 14 day calendars and 9 hours per day. The simulator randomly builds schedules with the desired density before running an experiment. Tasks from this schedule are introduced to the system at the desired rate. However, after some time, there are no more tasks left to be given to the system. The system is run until processing of all task requests are over. Therefore, the arrival rate is actually maintained over only a portion of each of the runs. The results are averaged over 1000 runs. All tasks are unconstrained in time, and hence can be scheduled at any interval on the calendar.

From the figures, it is obvious that the number of iterations to schedule a task increases both with the density of the calendar and with the task arrival rates. At low arrival rates, some iterations are saved with commitment, but these savings vanish with increasing arrival rates. However, commitment seems to considerably affect the performance of the system on the task hours missed metric. The effect is particularly pronounced at high densities, at which a much larger proportion of the desired tasks cannot be scheduled using the committed strategy.

We monitored the simulation to explain the observed behavior of the system with the different commitment strategies. In particular, we counted the following:

- number of instances when a resource was proposed for overlapping time intervals for two tasks and both could be scheduled in that interval (**possible** interaction),
- number of times a scheduler could not propose a particular interval because it overlapped with a blocked interval[2] (**preemptive** interaction).

[2]An interval can still be blocked using the **non-committed** commitment strategy when every

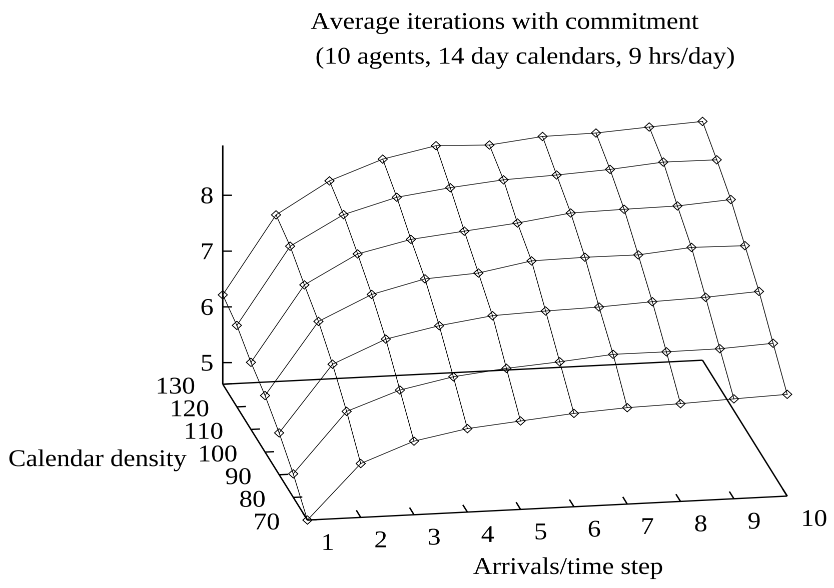

Figure 1: Effect of commitment strategy on the number of iterations with varying desired schedule density and arrival rate of tasks (10 agents, 14 day calendars, 9 hrs/day).

The former represents the frequency of backtrack required while using **committed** strategy, whereas the latter indicates the frequency with which the scheduling of one task constrains the available intervals for scheduling another task concurrently when commitment is used.

Experimental data suggests that the number of **preemptive** interactions per task is approximately 3 to 5 times more when the **committed** strategy is used when compared to the **non-committed** strategy. This demonstrates the introduction of severe constraints on the available intervals on which a task can be scheduled. For the **committed strategy**, the number of these conflicts per task range from 39 (task arrival rate 1, desired schedule density 70) to 109 (task arrival rate 10, desired schedule density 130). The corresponding numbers for the **non-committed** strategy are 7 and 23 respectively.

On the other hand, the number of **possible** interactions observed while using the **non-committed** strategy varies from only 2.5 to 3.1 per hundred tasks scheduled. Though no such conflicts can occur when using the **committed** strategy, the savings obtained by using the **committed** strategy is not significant. However, we believe this effect is reduced due to the unconstrained natures of tasks; if tasks are constrained to be scheduled in a small window on the calendar it may be beneficial

resource agent has proposed a common time interval, and the host agent uses a two-phase lock and commit mechanism to schedule the task.

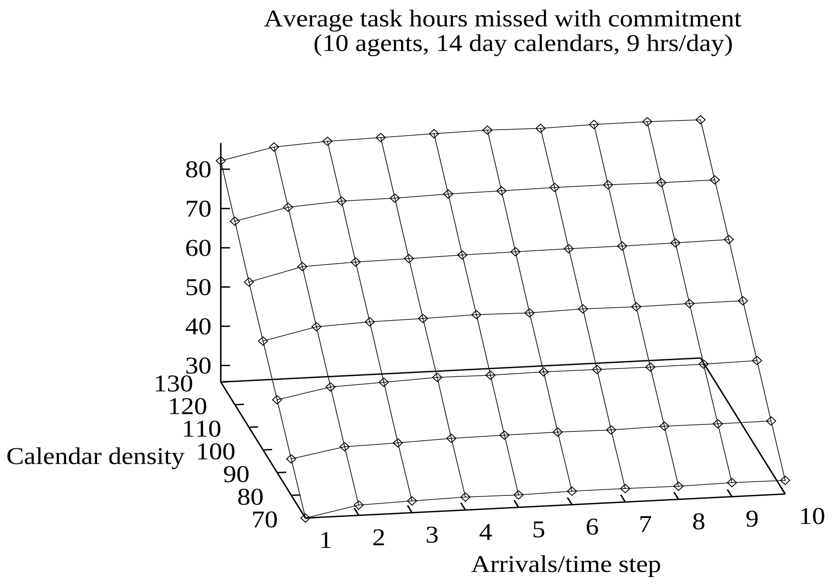

Figure 2: Effect of commitment strategy on task hours missed under varying desired schedule densities and task arrival rates (10 agents, 14 day calendars, 9 hrs/day).

to used the **committed** strategy.

We conjectured that commitment will be more useful when whatever being proposed had a good probability of being accepted. In order to substantiate this claim, we ran further experiments with a smaller organization (5 agents, each resource calendar being 5 days long). Task arrival rate was varied from 1 to 5 per time step, and desired schedule densities were varied from 5 to 35 hours (out of a maximum of 45 hours per resource). The **committed** strategy produced a smaller number of iterations to schedule a task over the entire range of these environmental conditions. This is so, because, with few agents and few tasks to schedule, resource calendars look similar, and hence intervals proposed for a task by one agent have a high likelihood of being accepted by others. Concurrent negotiation on two tasks using overlapping intervals can therefore lead to **possible** interactions, which can be avoided by using the **committed** strategy. The number of task hours that could not be scheduled by using the **committed** strategy is more than that when using the **non-committed** strategy, but both these values are negligible (at most 3.5% of the requested hours could not be scheduled).

From these initial experiments, it seems that commitment may be useful in a small number of particular situations. Hence, we should adopt a flexible commitment strategy that does not commit in most cases, but resort to commitment whenever beneficial. In the following, we identify the environmental conditions to be used by an adaptive commitment strategy.

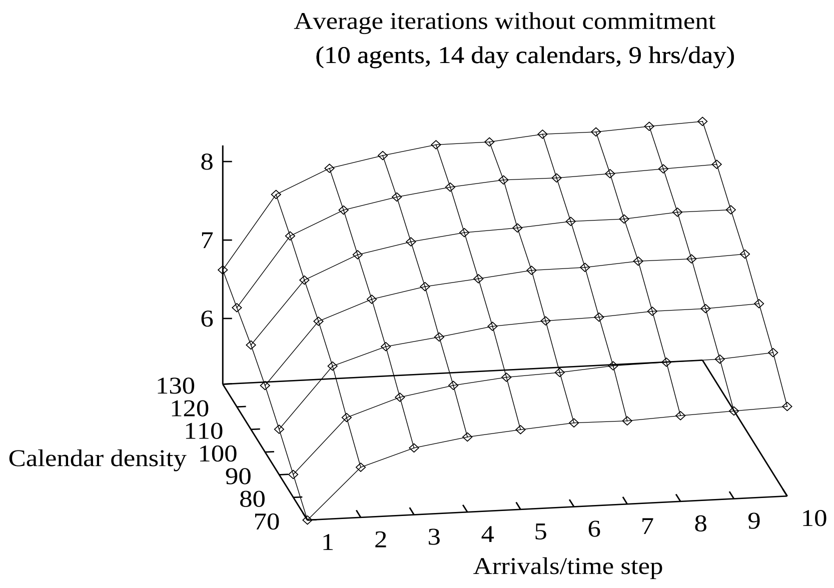

Figure 3: Effect of commitment strategy on the number of iterations with varying desired schedule densities and task arrival rates (10 agents, 14 day calendars, 9 hrs/day).

5. Adaptive scheduling

The need for an adaptive scheduling agent can hardly be overstated. Our experiments with the commitment strategy options shows that no one option is dominant over another in the sense that it performs better under all circumstances on all performance metrics. In most cases, superior performance on one performance metric is traded off against inferior performance on some other performance metric. More importantly, changing environmental factors like system load, organization size, etc., can produce a change in the option that will produce the best results on any given performance metric. We would like our automated scheduler to take advantage of any mechanism to predict the best strategy option for a given performance metric and given environmental and system conditions. Such a scheduler would be adaptive to changes in the system and the environment, providing us with the most desirable performance as measured by a given performance metric [12].

5.1. Environmental factors influencing commitment decisions

The following are some of the most important environmental factors that can be used to guide the choice of commitment strategies in an automated scheduling agent. For each of these, and the resultant matrix, we are describing qualitative intervals,

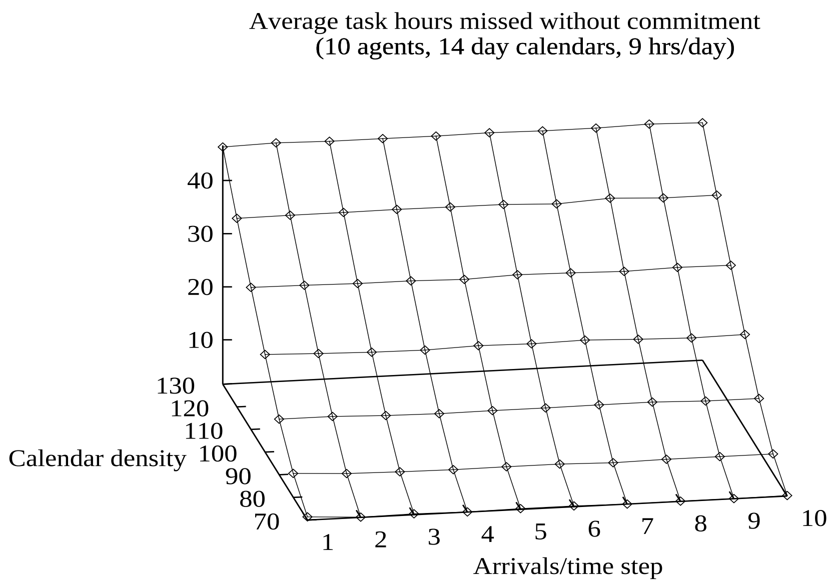

Figure 4: Effect of commitment strategy on task hours missed under varying desired schedule densities and task arrival rates (10 agents, 14 day calendars, 9 hrs/day).

although a more thorough quantitative analysis of the impact of these factors is possible [10].

Success probability for proposed interval: This factor measures the probability of success that the proposed interval will be free in the calendar of every resource manager, and hence can be used to schedule the task. We will consider cases of *high* and *low* success probability.

Window of acceptance: This factor is measured by the number of possible starting times for the task requested ($|S_i|$). We will consider tasks with *large* or *small* windows of acceptance.

Negotiation Density: The number of tasks being currently negotiated on in a given portion of the calendar will impact the likelihood of conflicts between processing these tasks. We will consider cases of *high* and *low* negotiation density.

5.2. Adaptive choice of Commitment Strategy

The choice matrix that enables us to choose the most appropriate commitment strategy when proposing for scheduling a given task is based on relevant environmental factors, and is presented in Table 1.

Success prob	High	High	High	High	Low	Low	Low	Low
Acceptance window	Large	Large	Small	Small	Large	Large	Small	Small
Neg. density	High	Low	High	Low	High	Low	High	Low
Comm. strategy	**NC**	**C**	**C**	**C**	**NC**	**NC**	**NC**	**C**

Table 1: Matrix to choose the most appropriate commitment strategy option (**C** for committed, **NC** for non-committed) given combination of environmental factors.

From the table, it is clear that the deciding factor for the commitment strategy choice is the success probability of scheduling the task using one proposal. An adaptive agent should commit to its proposed intervals whenever the likelihood of scheduling the task with that proposal is high. The only exceptions are when

(i) window of acceptance is large and negotiation density is high (other tasks vying for the same interval, and there are enough other possibilities to schedule the current task); do not commit.

(ii) window of acceptance is small and negotiation density is low (not many possibilities remaining for this task, and few other tasks are being considered for this part of the calendar); do commit.

The third combination of environmental factors (high success probability, small acceptance window, and high negotiation density) is a close call, and the scheduler may be well advised to consider the priorities of other tasks being negotiated in the same part of the calendar before making a choice of commitment: commit if the current task is of higher priority than others negotiated in the same part of the space, do not commit if the current task has a very low priority.

6. Conclusion

We have recognized commitment as a key mechanism for coordinating the activities of multiple cooperative agents. This is particularly true in dynamic domains and where agents work concurrently on interdependent activities to increase the throughput of the system. In particular, we have investigated the effect of two simple commitment strategies in the domain of distributed scheduling in which agents manage local resource calendars and must cooperate to bring together all the resources required to process any assigned task. We have identified organization size (number of agents and length of resource schedules), arrival rate of tasks, and acceptable window of scheduling tasks to be the relevant environmental factors determining the usefulness of alternate commitment strategies. We have formally represented the types of interactions between scheduling processes when using these commitment strategies, and presented experimental results to determine the relative frequencies of these interactions. These results suggest that commitment should be used sparingly, and we have outlined an adaptive commitment strategy that makes the most effective commitment decision based on key environmental factors.

The commitment strategies and the distributed scheduling protocol we have studied in this paper are useful for domains where tasks arrive over time, and require multiple distributed resources to be processed. Our approach helps to maintain local autonomy and privacy requirements without sacrificing the capability of concurrently processing multiple task requests. If all tasks are known ahead of time or new task arrivals are spaced far apart, number of resources required per task is very small, or resource managers agree to share all information without any restriction, a more traditional, centralized scheduling approach [7] should be preferred.

We would like to extend this work in two important directions:

(i) For the distributed scheduling domain, we want to investigate more opportunistic commitment strategies, where a proposed interval will be proposed for another task only if the utility to do so is higher. Utility of proposing an interval for a task can be calculated from the priority of the task, the success likelihood of scheduling the task using one proposal, and the remaining window of acceptance for the task. Using this approach, a proposed interval may appear committed or not committed from the point of view of a newly arrived task, depending on the relative utilities of proposing the interval for the new task and for the task for which the interval has already been proposed.

(ii) To address a more general problem, we intend to develop quantitative models of the effectiveness of commitment in cooperative planning. The goal is to enable agents to make effective decisions over what course of action to commit to, and at what stage in the negotiation process.

References

[1] Philip R. Cohen and Hector J. Levesque. Persistence, intention, and commitment. In *Proceedings of the 1986 Workshop on Reasoning About Actions and Plans*, July 1986.

[2] Susan E. Conry, Robert A. Meyer, and Victor R. Lesser. Multistage negotiation in distributed planning. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 367–384. Morgan Kaufman, 1988.

[3] Piotr J. Gmytrasiewicz and Jeffrey S. Rosenschein. The utility of embedded knowledge-oriented actions. In *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, pages 155–169, May 1993.

[4] Nick R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993.

[5] Henry F. Korth and Abraham Silberschatz. *Database System Concepts.* McGraw Hill, New York, NY, 1991.

[6] Frederick C. Mish, editor. *Webster's Ninth New Collegiate Dictionary*, Springield, Massachusetts, 1991. Merriam Webster.

[7] S.J. Noronha and V.V.S. Sarma. Knowledge-based approaches for scheduling problems: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 3(2):160–171, June 1991.

[8] James L. Peterson and Abraham Silberschatz. *Operating System Concepts*. Addison Wesley, Reading, MA, 1985.

[9] Thomas C. Schelling. *The Strategy of Conflict*. Havard University Press, 1960.

[10] Sandip Sen. *Predicting Tradeoffs in Contract-Based Distributed Scheduling*. PhD thesis, University of Michigan, October 1993.

[11] Sandip Sen and Edmund H. Durfee. Using temporal abstractions and cancellations for efficiency in automated meeting scheduling. In *Proc. of the International Conference on Intelligent and Cooperative Information Systems*, pages 163–172, May 1993.

[12] Sandip Sen and Edmund H. Durfee. On the design of an adaptive meeting scheduler. In *Proc. of the Tenth IEEE Conference on AI Applications*, pages 40–46, March 1994.

[13] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, December 1980.

[14] K. Sycara, S. Roth, N. Sadeh, and M. Fox. Distributed constrained heuristic search. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1446–1461, December 1991. (Special Issue on Distributed AI).