

A Negotiation-based Interface Between a Real-time Scheduler and a Decision-Maker *

Alan Garvey and Keith Decker and Victor Lesser
Computer Science Department
University of Massachusetts
Amherst, MA 01003

March 25, 1994

Abstract

Our real-time problem solver consists of a real-time scheduling component and a decision making component. These components are semi-autonomous and have a complex, bidirectional interface. We have found it useful to model this interface as a negotiation process, where conflicts between what each component wants to do are resolved through negotiation. A distinguishing feature of this work is that we argue that negotiation is useful, even within a single agent architecture.

1 Introduction

In real-time problem solving it is useful to separate real-time scheduling from decision making for reasons of efficiency, modularity and reusability. However, such a separation results in two, semi-autonomous components that need to work together to solve problems. Because they have different knowledge and different perspectives, these two components can disagree about what the agent should do. We have found that the interface between these components can usefully be modeled using negotiation. One example of a high level decision maker is a coordination module for an agent in a multiagent system that makes decisions about what information to communicate to other agents and when to communicate it. Note that here we are focusing on the negotiation between such a coordination module and a scheduler, but negotiation may also be useful between coordination modules of different agents.

As an example of what we mean by such an interface, consider a situation where multiple agents are working on a problem. Agent A has a method (Method A1) that enables the execution of an important method at another agent B. At this point the decision-maker at Agent A realizes that it should try to get the scheduler to schedule Method A1. It can do this by associating a *do commitment* with Method A1, meaning that it requests that the scheduler try to build schedules that execute Method A1. Agent A's scheduler returns a schedule that

*This material is based upon work supported by the National Science Foundation under Grant No. IRI-9208920, NSF contract CDA 8922572, DARPA under ONR contract N00014-92-J-1698 and ONR contract N00014-92-J-1450. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

completes executing Method A1 at time 7. The decision-maker at Agent A tells other decision-makers that it can commit to giving them the result of Method A1 at time 8 (allowing time for communication to occur). At the second agent B, the decision-maker receives this message and passes it along to the scheduler, which reports back that time 8 is too late—the result is needed by time 6. Agent A's decision maker is informed of this feedback, and Agent A again invokes its scheduler, now with a *deadline commitment* to complete Method A1 by time 5 (to allow time for communication). Agent A's scheduler returns a schedule that commits to completing Method A1 by time 5 and Agent A communicates this information to the other agent, which is now able to complete its method by the deadline.

We believe that the communication between the decision-maker and the scheduler can best be modeled as a process of *negotiation.* Negotiation is coordinated communication with the goal of enabling or improving problem solving. [Lâasri *et al.*, 1992] describe the information exchanged in negotiation in terms of *proposals*, *critiques*, and *explanations.* The description of the interface in this paper is arranged around these kinds of information. We first describe the basic input/output behavior of the scheduler in terms of proposals and explanations. We then examine feedback in the form of critiques, either of the schedule as produced by the scheduler or of the input specification. The paper concludes with a discussion of future work. Related papers of ours discuss the details of the scheduling [Garvey *et al.*, 1993, Garvey and Lesser, 1993] and how the coordination modules communicate with one another [Decker and Lesser, 1994].

2 Proposals and Explanations

Problem-solving begins when a problem to be solved arrives at the decision-maker. In our work, problems are presented to the decision-maker as TÆMS task structures. The form of such task structures is described in more detail in [Decker and Lesser, 1993]. Briefly, a problem episode E consists of a set of independent *task groups* $\mathbf{E} = \langle \mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n \rangle$, each with a hard deadline $\mathrm{D}(\mathcal{T})$ and containing interrelated *tasks* $T \in \mathcal{T}$. Within a task group, tasks form a directed acyclic graph through the subtask relationship. The *quality* or value of a task T at a particular time t (notated Q(T,t)) is a function of the quality of its subtasks (in this paper, the function is one of minimum (AND-like) or maximum (OR-like)). At the leaves of the DAG are *executable methods* M representing actual computations that can be performed. A single agent may have multiple methods for a task that trade-off time and quality. Besides the subtask relationship tasks can have other relationships to methods representing the interactions among tasks. Such relationships include enables(T, M, θ) meaning that the enabling task T must have quality above a threshold θ before the enabled method M can execute, facilitates(T, M, ϕ_d, ϕ_q) meaning that if the facilitating task T has quality above a threshold then the facilitated method M can execute more quickly (proportional to ϕ_d) and/or achieve higher quality (proportional to ϕ_q), and hinders(T, M, ϕ_d, ϕ_q) (the opposite of facilitates) where if the hindering task T has quality, then the hindered method M will achieve reduced quality and/or increased duration if it is executed. Note that these relationships occur from a task or method to a method. A relationship from Task A to Task B is translated to relationships from Task A to all methods below Task B.

Much of the discussion in the remainder of this paper is grounded in a multiagent example scenario. In the multiagent scenarios each executable method is executable by exactly one

agent, however several agents may have identical methods for achieving quality for the same task. The goal of the decision-makers is to work together to produce the highest possible quality for as many task groups as possible, i.e., each attempts to maximize the *global* utility measure $\mathrm{U}(\mathbf{E}) = \sum_{\mathcal{T} \in \mathbf{E}} Q(\mathcal{T}, \mathrm{D}(\mathcal{T}))$. This is not straightforward, because each agent sees only some part of the total task structure, and it may be the case that no agent sees the entire structure. Thus the decision maker cannot simply ask the scheduler to optimize this global criteria. One kind of information that agents can communicate to one another is information about the task structures that they see. The decision-makers are responsible for coordinating their activity so as to avoid redundant method execution and allow relationships that extend across agents to be exploited or avoided as appropriate. The role of the scheduler is to schedule execution of local methods according to criteria provided by the decision-maker.

2.1 Scheduler Inputs

The decision-maker *proposes* to the scheduler that a solution to each newly arriving task group be added to the schedule of methods to be executed. A basic request to the scheduler (its input) consists of four things: the task structures to be scheduled, a set of commitments, a set of non-local commitments, and a runtime indication.

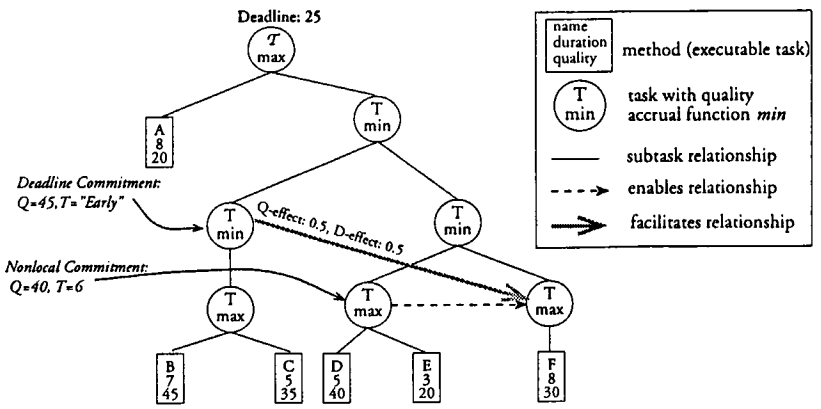

Figure 1: An example of a complete input specification to the scheduler.

The task structures to be scheduled $\mathbf{E}_S$, should include some indication of what aspects of those structures have changed since the scheduler was last invoked. If we write $B_A^t(X)$ to indicate what agent A believes at time t about X, then the scheduler at agent A at time t has access to $B_A^t(\mathbf{E})$, and $B_A^t(\mathbf{E}) \setminus B_A^{t-1}(\mathbf{E})$.

The set of commitments $\mathbf{C}$ are constraints that the invoker would like the scheduler to try to satisfy. We have defined three types of commitments:

- $C(\mathsf{Do}(T, q))$ is a commitment to 'do' (achieve quality for) T and is satisfied at the time t when $Q(T, t) \geq q$. A 'don't' commitment is also possible.
- $C(\mathsf{DL}(T, q, t_{dl}))$ is a 'deadline' commitment to do T by time t_{dl} and is satisfied at the time t when $[Q(T, t) \geq q] \wedge [t \leq t_{dl}]$. A $C(\mathsf{Do}(T, q))$ is really shorthand for $C(\mathsf{DL}(T, q, \mathrm{D}(T)))$.
- $C(\mathsf{EST}(T, q, t_{est}))$ is a 'earliest start time' commitment to not begin work on T before time t_{est} and is satisfied at the time t_{est} iff $\forall t \leq t_{est},\ Q(T, t) \leq q$.

The importance of local commitments such as these are as *soft* constraints on the possible solutions. Any scheduler that can schedule real-time method executions can already deal with

hard constraints such as deadlines and earliest start times. Soft commitments are needed to handle the coordination of multiple agents where there is more than one way to solve a task or where there are soft coordination relationships such as facilitates. When invoking the scheduler in a query mode, the decision-maker may also supply the symbolic values 'early' for a deadline commitment and 'late' for an earliest start time commitment, which indicates to the scheduler that it should attempt to satisfy the commitment as early or late as possible.

The set of non-local commitments NLC are commitments that the scheduler can assume will be satisfied. These are of the form of the commitments mentioned above and tell the scheduler to expect to receive the indicated results at the indicated time.

In multi-agent problems, non-local commitments can be used to communicate work that will be done by other agents. This component is necessary for achieving coordinated behavior in complex domains. These non-local commitments might be created at run time by the decision-makers, or they might be derived from pre-defined 'social laws' [Shoham and Tennenholtz, 1992] that all agents agree to, or are constructed to, satisfy. Another effect of NLCs in multi-agent problems is the triggering of non-local effects (coordination relationships); each non-local deadline commitment, for example, implies an earliest start time on the 'affected' end of any relationships. For hard relationships like enables this implies a hard earliest start time; for soft relationships it actually expands the search space (since each affected task can be started either before or after the earliest start time with different results).

Various mechanisms for controlling the runtime of the scheduler that we do not discuss here.

2.2 Basic Scheduling Algorithm

In general, these scheduling problems are NP-Hard. For that reason, heuristic scheduling is necessary for all but the smallest problem instances. We refer you to [Garvey *et al.*, 1993, Garvey and Lesser, 1993] for a detailed discussion of the scheduling algorithms that we use.

2.3 Scheduler Output

The output from the scheduler after an invocation should include at least one valid schedule, a list of satisfied commitments, a list of violated commitments with alternatives, an indication of tasks that should be scheduled but are not, and an indication of the value of each returned schedule with respect to some fixed set of criteria.

A set of valid schedules S is returned that do a satisfactory job of satisfying the problem given to the scheduler. An individual schedule $S \in \mathrm{S}$ consists of at least a set of methods and start times: $S = \{\langle M_1, t_1\rangle, \langle M_2, t_2\rangle, \ldots, \langle M_n, t_n\rangle\}$. This output is of course necessary, and forms the initial *proposal* in the negotiation process. The remaining items provide an explanation of this proposal.

The next three items returned (satisfied commitments, violated commitments with alternatives, and multicriteria schedule values) are not *necessary* for the scheduler to provide, because they can all be derived mathematically from the schedule itself and the set of non-local commitments. However, for practical implementations, the scheduler often has this information at hand, or can collect it during schedule generation, and it would be expensive to recompute.

The set of input commitments that are satisfied in a schedule Satisfied(S) is returned ($\forall S \in \mathrm{S}, \mathrm{Satisfied}(S) \subset \mathrm{C}$). If the scheduler supports symbolic local commitments like 'early' deadline commitments and 'late' earliest start time, then it must also supply an indication of

when the commitment is expected to be satisfied in the schedule $\text{SatTime}(C, S)$. For example, if $C_1 = \text{DL}(T, q, \text{'early'})$ and $C_1 \in \text{Satisfied}(S)$ then $\text{SatTime}(C, S) = \min t \leq \text{D}(T)$ s.t. $Q_{\text{est}}(T, t, S) \geq q$.

The set of input commitments that are violated in a schedule $\text{Violated}(S)$ is returned. For each violated commitment, a proposed modification to the commitment that the scheduler *is* able to satisfy ($\text{Alt}(C, S)$) is also returned. For earliest start time and deadline commitments this involves a proposed new time and/or minimum quality. For do/don't commitments this involves a recommended retraction or a reduced minimum quality value. For example, for a violated deadline commitment $C(\text{DL}(T, q, t_{dl})) \in \text{Violated}(S)$ the function $\text{Alt}(C, S)$ returns an alternative commitment $C(\text{DL}(T, q, t^*_{dl}))$ where $t^*_{dl} = \min t$ such that $Q(T, t) \geq q$ if such a t exists, or NIL otherwise.

The knowledge that certain commitments are satisfied or violated is absolutely necessary to the decision-maker that uses commitments, regardless of the domain.

An indication of the "value" of the schedules that were returned according to several objective criteria. Some of the objective functions that can be measured include the total quality for all scheduled task groups,[1] the number of task groups that do/do not complete before their deadline, the amount of slack time available in the schedule to allow easy scheduling of new tasks and/or allow time for tasks to take longer than expected to run, and the number (or weighted value) of the commitments that are not satisfied in the schedule.

Complex real problems invariably involve multiple evaluation criteria that must be balanced with one another; we view this balancing as the role of the decision-maker, and the scheduler attempts to maximize the current criteria, often returning multiple schedules (e.g., one that best satisfies each of the current criteria.) While the ability to evaluate a schedule with respect to certain criteria could be implemented outside of the scheduler, the ability to attempt to optimize certain criteria can only be placed in the scheduler.

A minimal list of tasks in the task structure that the schedule is not providing quality for but would need to have quality to allow their task group to achieve non-zero quality.

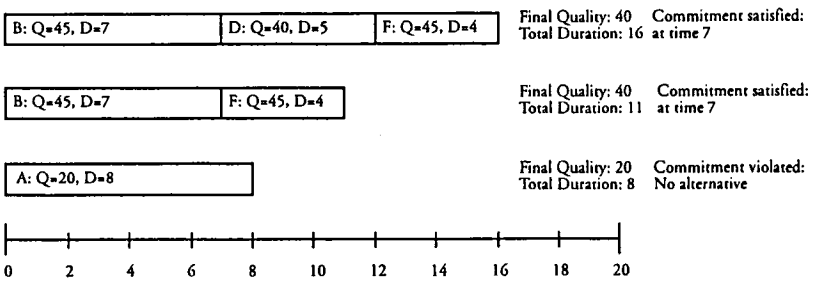

Figure 2: An example of the output of the scheduler for the example problem given above.

A summary of the output of the scheduler for the example problem given above is shown in Figure 2. In this example three schedules are returned. The bottom one produces a fast, low quality result, violating the given deadline commitment because no quality is ever generated for the committed task. The middle schedule produces the highest possible quality in the fastest possible time, satisfying the deadline commitment at time 7. The top schedule produces the highest quality possible completely locally, not relying on the given nonlocal commitment, also satisfying the deadline commitment at time 7. Which of these schedules is chosen by the

[1] This could be a weighted sum if task group importance varies, or some more complex function if desired.

decision-maker depends on the current evaluation criteria. If the fastest possible, acceptable result is desired, perhaps because of a large workload of other tasks, then the bottom schedule is chosen. If the best possible result in the minimum possible time is desired, the the middle schedule is chosen. If the best possible result that does not rely on other agents is desired (possibly because of other work that those agents need to do or a concern about the other agent's reliability) then the top schedule is chosen.

3 Critiques and Repair

Critiques are expressions of dissatisfaction with particular parts of a proposal. They can be made internally by the scheduler as part of the schedule creation process or externally by the decision-maker. In both cases the scheduler attempts to respond to the critique by *repairing* the schedule. Repair consists of deciding what part of the schedule to modify (not necessarily the exact part criticized by the critique) and deciding what modification to make.

At this point in our research, critiques are only generated internally by the scheduler. After the scheduler has generated a new schedule it critiques it looking for a few kinds of specific problems. If these problems are detected it attempts to repair them by adding new methods or idle time to the schedule. In the future we intend to add many more critiques and repairs to the scheduler, including critiques from outside the scheduler. Informally, we can consider a request from the decision maker to schedule a task 'early' as a critique (i.e., 'I like this schedule, but try to move this task up earlier in the execution order'). We also intend to study the tradeoffs associated with critiques, comparing the improvement in performance with the added runtime cost.

4 Example

We will now give a short example illustrating the ebb and flow of communication between the local scheduler and the decision maker in our implementation. Let the three classes of schedules typically returned by our scheduler be named by their primary evaluation criteria: BC (best committed), BQ (best quality), and BF (best finish time)—remember that the local scheduler is heuristic and cannot optimize these criteria. Initially the local scheduler will propose a set of schedules without any local or non-local commitments (BC = BQ). The decision-maker uses the BQ schedule as a starting point, looking at the *potential* of various activities to help or interfere with activities at other agents. For example, Task A at Agent 1 may be in the current BQ schedule, and be believed to enable tasks at Agent 2. Agent 1 can then *locally* commit to doing task A, asking the local scheduler to schedule it as early as possible (effectively, a critique). The local scheduler then returns a new set of possible schedules and their evaluations. If in the decision-maker's eyes Task A finishes early enough, it may make a *non-local commitment* to Agent 2 about the completion deadline for Task A. Agent 1 may do this several times, creating several local commitments and possible non-local commitments. The local scheduler may not always be able to make the proposed local commitment, or making the new commitment can cause older commitments to be violated. In this case the decision maker will often jettison the new commitment (which, because it has not been communicated to other agents yet, is painlessly retracted). Other agents will also be doing similar scheduling activities, so non-local commitments will be coming in from them. At some point a new task group may arrive, and Agent 1 may not have enough computational resources (or the scheduler may not be able to

find a schedule) to keep all commitments and produce the best quality schedule (BC $\neq$ BQ). At this point the decision maker has several choices: it can choose from one of the existing schedules or alter some set of local commitments and try to reschedule. In our implementation the decision maker always chooses from the existing set of schedules, and chooses the one with the best *global* utility in the decision maker's view. Global utility may differ from local quality because the local scheduler does not have a complete view of the problem[2]. After choosing a schedule, the decision-maker issues retractions for local commitments that have been violated, and new commitments when the violated commitment has an alternative. Thus the local scheduler and decision-maker carry on a fairly complex back-and-forth dialog, each bringing their unique viewpoints to the control of an individual agent.

5 Conclusions

We have presented the details of a complex, bidirectional interface between a decision maker and a scheduler. The major ideas presented here, including the details of the interface, have been implemented in a multi-agent problem-solver [Decker and Lesser, 1994]. The interface was developed in response to the real requirements of building this complex problem-solver to solve randomly generated task structures in the TÆMS environment. We have found the schedules produced and runtime of our scheduling algorithm to be acceptable for this problem-solver, but other applications might require more complex schedulers or faster schedulers.

How should we evaluate the interface? We have empirical evaluations of the system as a whole (multiple agents, each with a decision-maker and a local scheduler solving shared randomly generated problems), and of the scheduler in isolation, but neither of these evaluations cuts to the heart of the interface between the decision-maker and local scheduler in a single agent. What we have tried to do is argue for the *necessity* of various parts of the interface (i.e. problem specification, local and non-local commitments, a *set* of valid output schedules) and the efficiency of providing some other useful information (i.e. violations, alternatives, evaluation, tasks needed but without quality, runtime controls).

In the future we would like to extend the system in a few directions. One interesting area to explore is doing asynchronous, concurrent decision-making and scheduling. Both activities could go on simultaneously with both systems evaluating what requests to respond to first and how to respond to changing environments. Another area we intend to investigate is the effect of uncertainty in the duration and quality of methods. Such uncertainty increases the difficulty of scheduling and potentially reduces the reliability of commitments.

References

[Decker and Lesser, 1993] Keith S. Decker and Victor R. Lesser. Quantitative modeling of complex computational task environments. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 217–224, Washington, July 1993.

[2]This is similar to the way decisions can be made under different viewpoints in concurrent engineering systems: locally, I might prefer alternative A to alternative B, but if you say alternative B is unworkable or impossible from your viewpoint, I can make a decsion to pursue A even without being able to derive that result locally. This is another reason why we separate local scheduling from decision-making.

[Decker and Lesser, 1994] Keith S. Decker and Victor R. Lesser. Designing a family of coordination algorithms. Computer Science Technical Report 84-14, University of Massachusetts, 1994.

[Garvey and Lesser, 1993] Alan Garvey and Victor Lesser. Design-to-time real-time scheduling. *IEEE Transactions on Systems, Man and Cybernetics*, 23(6):1491–1502, 1993.

[Garvey *et al.*, 1993] Alan Garvey, Marty Humphrey, and Victor Lesser. Task interdependencies in design-to-time real-time scheduling. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 580–585, Washington, D.C., July 1993.

[Lâasri *et al.*, 1992] Brigitte Lâasri, Hassan Lâasri, Susan Lander, and Victor Lesser. A generic model for intelligent negotiating agents. *International Journal on Intelligent Cooperative Information Systems*, 1(2):291–317, 1992.

[Shoham and Tennenholtz, 1992] Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agnet societies (preliminary report). In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 276–281, San Jose, July 1992.