Exploiting Meta-Level Information in a Distributed Scheduling System*

Daniel E. Neiman, David W. Hildum, Victor R. Lesser
Tuomas W. Sandholm
Computer Science Department
University of Massachusetts
Amherst, MA 01003
DANN@CS.UMASS.EDU

May 13, 1994

Abstract

In this paper, we study the problem of achieving efficient interaction in a distributed scheduling system whose scheduling agents may borrow resources from one another. Specifically, we expand on Sycara's use of resource texture measures in a distributed scheduling system with a central resource monitor for each resource type and apply it to the decentralized case. We show how analysis of the abstracted resource requirements of remote agents can guide an agent's choice of local scheduling activities not only in determining local constraint tightness, but also in identifying activities that reduce global uncertainty. We also exploit meta-level information to allow the scheduling agents to make reasoned decisions about when to attempt to solve impasses locally through backtracking and constraint relaxation and when to request resources from remote agents. Finally, we describe the current state of negotiation in our system and discuss plans for integrating a more sophisticated cost model into the negotiation protocol. This work is presented in the context of the Distributed Airport Resource Management System, a multi-agent system for solving airport ground service scheduling problems.

*This work was partly supported by DARPA contract N00014-92-J-1698 and NSF contracts CDA-8922572 and IRI-9208920. The content of this paper does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

1 Introduction

The problem of scheduling resources and activities is known to be extremely challenging [8, 7, 14, 11]. The complexity increases when the scheduling process becomes dependent upon the activities of other concurrent schedulers. Such interactions between scheduling agents arise when, for example, agents must borrow resources from other agents in order to resolve local impasses or improve the quality of a local solution. Distributed scheduling applications are not uncommon, for example, the classic meeting planning problem [13] can be considered as a distributed scheduling problem; the airport ground service scheduling (AGSS) problem we address in this paper is another; and similar problems may arise in factory floor manufacturing domains.

In distributed scheduling systems, problem-solving costs will likely increase because of the interaction among agents caused by the lending of resources. One method of increasing the quality of solutions developed by such multi-agent schedulers and minimizing the costs of backtracking is to allow agents to communicate abstracted versions of their resource requirements and capabilities to other agents. The use of this *meta-level* information allows the scheduling agents to develop models of potential interactions between their scheduling processes and those of other agents, where an interaction is defined as a time window in which the borrowing or lending of a resource might occur. We show how the identification of interactions affects the choice of scheduling heuristics, communication, and negotiation policies in a distributed scheduling system. We discuss our heuristics in the context of a specific testbed application, the Distributed Airport Resource Management System (DIS-ARM).

2 Related Work

The use of meta-level information to define the interactions between agents has been studied extensively by Durfee and Lesser via the use of partial global plans [5]. This work has been extended by Decker and Lesser [3, 4] to incorporate more sophisticated coordination relationships. According to this framework, we can view our detection of potential loan requests using texture measures to be an identification of *facilitating* relationships, and our modification of the scheduling algorithm as an attempt to exploit this perceived relationship. The formulation of distributed constraint satisfaction problems as distributed AI was described by Yakoo [16], however, this work concentrated more on the problem of distributed backtracking rather than on coordinating agents.

The problem of coordinating distributed schedulers has been studied extensively by Sycara and colleagues [15]. They describe a mechanism for transmitting abstractions of resource requirements (*textures*) between agents. Each agent uses these texture measures to form a model of the aggregate system demand for resources. This model is used to allocate resources using various heuristics. For example, a *least-constraining-value* heuristic is used to allocate resources based on the minimization of the probablity that the reservation would conflict with any other. For each type of resource, one agent is assigned the task of coordinating allocations and determining whether requests can be satisfied. All resources of a given type are considered interchangeable and the centralized resource monitor does not need to perform significant planning to choose the most suitable resource; instead, its role is simply to ensure that each resource is allocated to no more agents than can be served by that resource during any given time period.

We investigate a similar use of abstracted resource demands for a case in which centralized resource monitors are not possible since resources of the same type may possess unique characteristics, and agents possess proprietary information about local resources (such as current location and readiness). Agents may respond to a request for a resource either by immediately satisfying it with a reservation, denying it, or by performing local problem-solving actions to attempt to produce a suitable reservation.

In our domain, we have found that Sycara's texture measures alone do not convey sufficient information to allow satisfactory scheduling. Their texture measures consist of a demand profile for each resource which represents, for each time interval, the sum of probabilities that resource requests will overlap that interval. These probabilities are based on the assumption that reservations can occur at any time within the requested interval. Assignment of resources is then performed using these probabilities to implement a least-constraining-value heuristic.

These texture measures do not capture sufficient information regarding time-shift preferences of resource assignments within the specified interval. In our domain, resources may legally be assigned at any time within the interval between the earliest start time and the latest finish time, but for some activities, there exist strong preferences as to which end of the interval the assignment is biased. For example, when scheduling ground services for an airport, once a flight arrives, it is important to unload baggage as early as possible so that necessary transfers can be made to connecting flights. The shift preference can be determined by the assigning agent using domain knowledge, provided that it knows the nature of the task generating the request. Because this information is not captured in the texture measures, the heuristic described by Sycara, *et al.* is likely to lead to poor schedules within the airport ground service scheduling domain.

3 Overview: The Distributed Dynamic Scheduling System

In order to test our approach to solving distributed *resource-constrained scheduling problems* (RCSPs), we have designed a distributed version of a reactive, knowledge-based scheduling system called DSS (the Dynamic Scheduling System) [9]. DSS provides a foundation for representing a wide variety of real-world RCSPs. Its flexible scheduling approach is capable of reactively producing quality schedules within dynamic environments that exhibit unpredictable resource and order behavior. Additionally, DSS is equipped to manage the scheduling of shared tasks connecting otherwise separate orders, and handle RCSPs that involve mobile resources with significant travel requirements.

DSS is implemented as an agenda-based blackboard system [6, 1] using GBB (the Generic Blackboard System) [2]. It maintains a blackboard structure upon which a developing schedule is constructed, and where the sets of orders and resources for a particular RCSP are stored. A group of knowledge sources are provided for securing the necessary resource reservations. These knowledge sources are triggered as the result of developments on the blackboard, namely the creation and modification of the service goals attached to all resource-requiring tasks. Triggered knowledge sources are placed onto an agenda and executed in the order of their priority.

The Distributed Dynamic Scheduling System (DIS-DSS) maintains separate blackboard structures for each agent and provides communication utilities for transmitting requests and meta-level information between agents. Remote analogues of service goals, task structures, and other scheduling entities are created as needed to model the state of other agents. The information about other agents' schedules and commitments is incomplete and is limited to the content of goals, meta-level information, and those parts of the schedule to which the local agent itself has contributed.

The approach we have taken towards distributing DSS is to view each agent as representing an autonomous organization possessing its own resources. It is this autonomous nature of the organizations that is the rationale for distributing the resource allocation problem. Although a centralized architecture might produce more efficient solutions, real world considerations such as cost and ownership often lead to confederations in which information transfer regarding commitments and capabilities is limited. In this model, the primary relationship between agents is a commitment to exchange resources as needed and a willingness to negotiate with other agents to resolve impasses. This model of a decentralized group of agents performing independent tasks in a resource-constrained environment is similar to the architecture of Moehlman's Distributed Fireboss [10]. We distinguish our work from Moehlman's by our use of meta-level information to control the decision process by which agents choose to resolve impasses locally, through backtracking and constraint relaxation, or through requests to remote agents.

Because resources are owned by specific agents and possess unique characteristics regarding location and travel times that are known only to the owning agent, we can not define central resource monitors responsible for allocating each type of resource. This, again, distinguishes our approach from that of Sycara, et al. [15]. Agents requiring a resource must communicate directly with the agent owning a resource of that type and negotiate for its loan.

This architecture provides a rich domain for the study of agent coordination issues in a distributed environment; agents must be able to model the interactions of their tasks with those of neighboring agents closely enough to be able to determine which agents will be most likely to provide the desired resources at the lowest cost to both agents. This coordination requires local reasoning on the part of agents in order to determine how to cooperate efficiently with an acceptable level of communication and redundant computation.

3.0.1 Assumptions

In our work with DIS-DSS, we have made a number of assumptions about the nature of agents, schedules, and communication overheads.

- Agents are cooperative and will lend a resource if it is available.
- Agents will only request a resource from one agent at a time – this is to avoid the possibility of redundant computation and communication if multiple agents attempt to provide the resource cf. [12].
- Once agents have lent a resource to another agent, they will never renege on this agreement. This limits the ability of the system to perform global backtracking; we intend to eliminate this restriction in the next version of the system.
- Communication is asynchronous and can occur at any point during the construction of a local schedule; therefore requests may arrive before an agent has completely determined its own requirements for resources in the time window of interest.
- The cost of messages is largely in the processing and in the inherent delay caused by transmission – the amount of data within the message may be large, within limits.

3.0.2 Communication of Abstract Resource Profiles

Without information regarding other agents' abilities to supply missing resources, an agent may be unable to complete a solution, or may be forced to compromise the quality of its solution. To allow agents to construct a model of global system constraints and capabilities, we have developed a protocol for the exchange and updating of resource profiles: summarizations of the agent's committed resources, available resources, and estimated future demand.

Upon startup, each agent in DIS-DSS receives a set of orders to be processed. The agents examine these orders and generate an abstract description of their resource requirements for the scheduling period. This *bottleneck-status-list* consists of a list of intervals, with each interval annotated by a triple: resources in use, resources requested, and resources available. The request field of this triplet represents an abstraction of the agent's true resource requirements. Certain aspects of a reservation such as mobile resource travel times to the objects to be serviced, cannot be easily estimated in advance. The time intervals specified for each resource request are pessimistic, consisting of the earliest possible start time and latest possible finish times for the activity requesting that resource. The true duration of the task can be estimated by the scheduling agent using its domain knowledge regarding the typical time required to perform a task. We define the demand for a resource r performing task T in interval (t_j, t_k) to be:

$$\text{avg\_demand}(T, r, t_j, t_k) = \text{duration}(T, r)/(t_k - t_j)$$

Once resource abstractions have been developed for each resource type required (or possessed) by the agent, it transmits its abstractions to all other agents. Likewise, it receives abstractions from all agents. Once the agent has received communications from all other agents, it prepares a map of global resource requirements and uses it to generate a set of data structures called *lending possibilities*. Each lending possibility represents an interval in which some agent appears to have a shortfall in a resource. For each lending possiblity, the agent generates a list of possible lenders for that resource, based on the global resource map and its knowledge of its own resource requirements. These lending possibility structures are used to predict when remote agents may request resources and when the local agent may need to borrow resources. This information guides the agent's decision-making process in determining both when to process local goals and when and from whom to request resources.

3.1 The Distributed Airport Resource Management System

The Distributed Airport Research Management System testbed was constructed using DIS-DSS to study the roles of coordination and negotiation in a distributed problem-solver. DIS-ARM solves distributed AGSS problems where the function of each scheduling agent is to ensure that each flight for which it is responsible receives the ground servicing (gate assignment, baggage handling, catering, fuel, cleaning, etc.) that it requires in time to meet its arrival and departure deadlines. The supplying of a resource is usually a multi-step task consisting of setup, travel, and servicing actions. Each resource task is a subtask of the airplane servicing supertask. There is considerable parallelism in the task structure: many tasks can be done simultaneously. However, the choice of certain resource assignments can often constrain the start and end times of other tasks. For example, selection of a specific arrival gate for a plane may limit the choice of servicing vehicles due to transit time from their previous servicing locations and may limit refueling options due to the presence or lack of underground fuel tanks at that gate. For this reason, all resources of a specific type can not be considered interchangeable in the AGSS domain. Only the agent that owns the resource can identify all the current constraints on that resource and decide whether or not it can be allocated to meet a specific demand.

4 Exploiting Meta-level Information in DIS-DSS

In this section, we examine three areas in which meta-level abstractions of global resource requirements are exploited in DIS-DSS. We show how the goal rating scheme of an agent's blackboard-based scheduler is modified to satisfy the twin aims of scheduling based on global constraints and of planning activities in order to reduce uncertainty about agent interactions. We describe how communication of resource abstractions is based on models of agents' interests and the manner in which agents choose between local and remote methods of satisfying a request.

4.1 Scheduling using Texture Measures

Many scheduling systems divide processing into the categories of variable selection, the choice of the next activity to schedule, and value selection, the selection of a resource and time slot for that activity. In DIS-DSS, variable selection corresponds to the satisfaction of a particular resource request. Value selection is handled in DSS by a collection of opportunistic scheduling heuristics. We focus here on the problem of coordinating resource requests so that local variable-selection heuristics possess sufficient information to make informed decisions.

In many knowledge-based scheduling systems, the object of control is to arrange scheduling activities so that the most tightly constrained activities are scheduled first in order to reduce the need for backtracking. In a distributed system, we have an additional criterion: to schedule problem-solving activities in such a way that global uncertainty about certain tasks is reduced before decisions regarding those tasks are made. A scheduler may be uncertain of whether other agents will request a resource in a tightly constrained time period and whether other agents will be able to supply a needed resource. While the resource abstractions may indicate a loan request is likely, the duration of the loan and details of the resource's destination can only be determined once the request has been received. Likewise, details of the precise timing and duration of a loan can only be determined upon receipt of a remote reservation. We have added coordination heuristics to the agenda scheduler of DIS-DSS whose purpose is to promote problematic activities in each agent's scheduling queue so that their early execution will reduce uncertainty about global system requirements.

In the DIS-DSS blackboard-based architecture, tasks which require resources generate *service-goals*. Requests received from remote agents generate *remote-service-goals*. Each goal stimulates knowledge sources that act to secure an appropriate resource. The order of execution of knowledge sources depends on the rating of the stimulating goals. Goals are rated using a basic 'most-tightly-constrained-first' opportunistic heuristic. The goals are then stratified according to the following scheme, with the uppermost levels receiving the highest priority and contention within each level being resolved according to the basic rating heuristic.

1. Tightly constrained goals that may not be satisfiable locally or that can only be satisfied by a borrowing event *and* remote requests that do not overlap any local request.
2. Tightly constrained goals that can *only* be satisfied locally.
3. Goals representing requests from remote agents that overlap local goals.
4. Unconstrained or loosely constrained tasks.
5. Goals that *potentially* overlap with tasks of remote agents.

A goal g is considered to be tightly constrained in interval (t_j, t_k) if there exists a time within that interval such that for each resource type r that can satisfy g, the number of unreserved resources is less than the sum of the average demand for all outstanding goals.

$$\forall\ r\ s.t.\ \mathrm{Sat}(g, r)\ \exists\ t \in (t_j, t_k)\ s.t.$$

$$\mathrm{N}_{available}(r, t) < \sum_{g} \mathrm{avg\_demand}(task(g), r, t_j, t_k)$$

A goal potentially overlaps with a task of a remote agent if there exists a lending-possibility data structure for that remote agent describing a potential shortfall within the time interval spanned by that goal for some resource type that could satisfy the goal.

The rationale for this goal ordering is as follows. Goals that can not be satisfied locally must be transmitted to remote agents. The transmission of a goal conveys considerably more information than is available in the resource texture profiles. The potential lending agent will therefore have more accurate information regarding the interval for which the resource is desired and the preferred shift preference for the reservation in that interval (early or late). Once it has received the goal, it will be able to make more informed decisions about the tightness of constraints for both the local and remote goals. If the agent is able to satisfy the remote goal, it will be able to update its resource demand curve and transmit it to other agents who may also have been potential lenders of that resource. For all these reasons, early transmittal and satisfaction of remote service goals is desirable.

Tightly constrained goals that potentially overlap remote requests are deferred until some overlapping goal arrives, or until a resource update arrives indicating that the remote agent no longer requires that resource, or until no other work is available for the agent to perform. By deferring goals until more information about interactions is available, the system can avoid making premature decisions while at the same time working on unrelated or less constrained tasks. Once a request arrives, conflicts for resources can be arbitrated according to which goal is most pressing and least conducive to backtracking and/or constraint relaxation.

There are a number of competing requirements for the rating and processing of remote service goals. One would like to process a remote service goal as soon as possible in order to return information to the requesting agent. At the same time, both local and remote service goals requesting the same type of resource should be rated according to the same constraint tightness heuristics. The goal rating function in DIS-DSS attempts to satisfy these requirements by prioritizing those remote service goals that do not overlap any local service goals and by mapping overlapping remote service goals onto the same priority level as those local goals that they overlap. Note that the "overlapping" relationship is transitive: if the priority of a goal is reduced while waiting for a remote request, any lower rated goal that overlaps that goal's time interval must also wait even though it may not directly overlap the interval of the potential remote request.

4.2 Guiding Communication using Texture Measures

Reducing communication costs is an important issue in distributed systems. For this reason, DIS-DSS agents use the lending possibility models of agent interactions to guide communication activities. When its resource requirements change, an agent transmits the information about the resource type only to those agents who, based on its local information, would be interested in receiving updates concerning that resource type. An agent with no surplus resources of a given type may not be interested if the local agent increases its need for a particular resource, likewise, an agent with a surplus of a particular resource may not need to be notified if an agent reduces its demand for that resource type. However, agents who possess shortfalls in a time interval for a particular type of resource will receive updates during processing whenever an agent increases the precision of its resource abstractions by securing or releasing a resource.

The use of local knowledge to guide communication episodes may lead to agents' knowledge of the global state of the system becoming increasingly out of date. The degree to which this should be allowed to happen is dependent upon the acceptable level of uncertainty in the system and the accuracy with which resource abstractions can be made.

4.3 Ordering Methods for Achieving Resource Assignments

In DSS, the process of securing a resource is achieved through a series of increasingly costly methods: assignment, preemption, and right shifting. These correspond roughly to request satisfaction, backtracking, and constraint relaxation. Preemption is a conservative form of backtracking in which existing reservations are preempted in favor of a more constrained task. Right shifting satisfies otherwise intractable requests by shifting the time interval of the reservation downstream (later) until a suitable resource becomes available. Because this method relaxes the latest finish time constraint, it has the potential to seriously decrease the quality of a solution. In the AGSS domain, for example, right shifting a reservation may result in late departures.

In DSS, methods are ordered according to increasing cost. In the distributed version of the system, the choice and ordering of methods is more complex. When an agent cannot immediately acquire a resource locally, it faces a decision: should it perform backtracking or constraint relaxation locally, communicating only when it has exhausted all local alternatives, or should it immediately attempt to borrow the resource from another agent? The decision-making process becomes even more difficult if we allow requests from remote agents to take precedence over local requirements such that agents may have to perform backtracking or constraint relaxation in order to satisfy a remote request. We consider this last decision process a form of *negotiation*, because it involves determining which of two agents should bear the cost of reduced solution quality and/or increased problem-solving effort.

In DIS-DSS, we use the lending possibility data structures to dynamically generate plans for achieving each resource assignment. When it appears that a remote agent will have surplus resources at the necessary time, then the agent will generate a request as soon as it becomes clear that the resource can not be secured locally. If, however, it appears that the resource is tightly constrained globally, the agent will choose to perform backtracking and/or constraint relaxation operations locally rather than engage in communication episodes that will probably prove futile.

One use of meta-information occurs during the planning for constraint relaxation. The scheduling agent attempts to minimize the magnitude of the right shift in order to reduce the effect of the constraint relaxation on the quality of the solution. To do this, the agent must determine whether the minimum right shift can be achieved locally or remotely. However, requiring agents to submit bids detailing their earliest reservations for a given resource would be a costly process. Instead, the agent uses the abstractions of remote resource availability to generate a threshold value for the right shift delay. If this value is less than the delay achieved through right shifting locally, the agent sequentially transmits the resource request to the appropriate remote agents. If a remote agent can provide a reservation with a delay of less than or equal to the threshold value, it immediately secures the resource. Otherwise, it returns the delay of the earliest possible reservation. If no reservation is found, the local agent sets the threshold to the earliest possible value returned by some remote agent. This new threshold is then compared to the current best local delay (which might have changed due to local scheduling while the remote requests were being processed). This process continues until a reservation is made or until the threshold becomes greater than the delay achievable by right shifting locally. Obviously, the better the initial estimate for the delay threshold, the less communication activities will be required.

The meta-information is also used to determine the order in which agents should be asked for resources, beginning with the agent(s) with the least tightly constrained resources.

5 Experimental Results

The performance of the mechanisms that we have developed for DIS-DSS were tested in a series of experiments using a single agent system as a basis for comparison. We used six scenarios designed to test the performance of the system in tightly constrained situations. The number of orders in each scenario ranged from 10 to 60 and a minimal set of resources was defined for each scenario. Each scenario was distributed for a three agent case. Orders were assigned to each agent on a round-robin basis such that each agent would perform approximately the same amount of work. Resources were distributed randomly so that in some cases each agent would possess all necessary resources while in other cases, borrowing from remote agents would be necessary.

We ran DIS-ARM on each scheduling scenario using the following configurations of the scheduler:

- The baseline case with a single agent.
- The 3 agent case with no use of meta-level information, and an opportunistic (most-tightly-constrained-variable-first) goal rating scheme
- The 3 agent case using the heuristic goal rating scheme incorporating meta-level information but requesting resources from remote agents only when all local methods have failed.
- The 3 agent case using heuristic goal rating, meta-level information, and dynamic reordering of resource acquisition methods to account for the probability of securing a goal either locally or remotely.

For each run, we recorded the average tardiness of the schedule, the number of failed goals (if any), the number of resource-securing methods tried, the number of requests, the number of satisfied remote service goals, and the number of communication episodes that occurred during problem solving. In each case, we assumed that communication costs were negligible in relation to problem-solving and that requests and resource constraint updates would be received on the simulation cycle immediately succeeding the one in which they were sent.

Because of the small number of test cases we have examined in our preliminary experiments, we present our results anecdotally. As expected, the distributed version of the scheduler always produces a schedule of somewhat lower quality than the centralized one. When the opportunistic scheduler of the centralized version is used for scheduling in a distributed environment, its lack of information about global constraints causes it to produce somewhat inferior results. The heuristic incorporating meta-level information consistently outperforms the opportunistic scheduler in terms of the number of tardy tasks. The opportunistic scheduler occasionally will produce a schedule with less total tardiness than the distributed algorithm. We interpret this as a trade-off between satisfying global requirements (by delaying certain goal satisfactions until remote information becomes available) and satisfying local requirements by producing needed results promptly. This is an interesting trade-off that we intend to study in depth. Attempting to always solve problems locally using preemption and constraint relaxation produced schedules with much greater delays than when agents dynamically determined when to request resources remotely based on the meta-level resource abstractions.

6 Conclusions and Future Work

The work we have performed with DIS-DSS is preliminary, but promising. Our results indicate that the idea of using meta-level information to schedule activities in order to reduce local uncertainty about global constraints results in better coordination between agents with a subsequent increase in goal satisfaction. We have also demonstrated that meta-level information can be successfully used to guide the choice between satisfying goals locally and remotely, and in optimizing the choice of agents from which to request resources.

Our experiments were performed with each agent's orders being defined statically before scheduling. This allowed the agents to develop a model of their predicted resource requirements before scheduling began. If we were to model a system in which orders changed dynamically, either due to equipment failures or timetable changes, we would expect the model of global resource requirements to become increasingly inaccurate. We would like to understand the implications of allowing jobs to arrive dynamically on the performance of a distributed system using meta-level information.

As well as continuing to explore the role of meta-level resource abstractions, we plan to use the DIS-DSS testbed to explore a number of important issues in distributed scheduling. One of our primary goals is to expand the idea of negotiation between agents that we have touched upon in this paper. Because the airport ground service scheduling domain represents a "real world" scenario, we are able to create a meaningful cost model involving not only the delay in each schedule, but the probable cost of that delay in terms of missed connections. By allowing agents to exchange this information when requesting resources, they will be able to more meaningfully weigh the importance of local tasks against the quality of the global solution.

References

[1] N. Carver and V. Lesser. The evolution of blackboard control architectures,. *Expert Systems with Applications- Special Issue on the Blackboard Paradigm and Its Applications*, 7(1):1–30, Jan–Mar 1994.

[2] D. D. Corkill, K. Q. Gallagher, and K. E. Murray. GBB: A generic blackboard development system. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 1008–1014, Philadelphia, PA., August 1986.

[3] Keith S. Decker and Victor R. Lesser. Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems*, 1(2):319–346, June 1992.

[4] 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.

[5] E.H. Durfee and V.R. Lesser. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1167–1183, September 1991.

[6] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. The hearsay-ii speech-understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213–253, June 1980.

[7] Mark S. Fox. *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*. PhD thesis, Carnegie Mellon University, Pittsburgh PA, December 1983.

[8] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, New York, 1979.

[9] David W. Hildum. *Flexibility in a Knowledge-Based System for Solving Dynamic Resource-Constrained Scheduling Problems*. PhD thesis, Computer Science Dept., University of Massachusetts, Amherst, MA 01003, May 1994.

[10] Theresa A. Moehlman, Victor R. Lesser, and Brandon L. Buteau. Decentralized negotiation: An approach to the distributed planning problem. *Group Decision and Negotiation*, (2):161–191, 1992.

[11] Norman Sadeh. *Look-Ahead Techniques for Micro-Opportunistic Job Shop Scheduling*. PhD thesis, Carnegie Mellon University, Pittsburgh PA, March 1991.

[12] Tuomas Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 256–262, Washington, July 1993.

[13] Sandip Sen and Edmund Durfee. A formal analysis of communication and commitment in distributed meeting scheduling. In *Proceedings of the Twelfth Workshop on Distributed AI*, Hidden Valley, PA, May 1993.

[14] Stephen F. Smith, Mark S. Fox, and Peng Si Ow. Constructing and maintaining detailed production plans: Investigations into the development of knowledge-based factory scheduling systems. *AI Magazine*, 7(4):45–61, Fall 1986.

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

[16] M. Yakoo, T. Ishida, and K. Kuwabara. Distributed constraint satisfaction for DAI problems. In *Proceedings of the 10th International Workshop on Distributed Artificial Intelligence*, October 1990.