Unsupervised Surrogate Agents and Search Bias Change in Flexible Distributed Scheduling

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

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

Abstract

Computational infrastructures for cooperative work should contain embedded agents for handling many routine tasks (Galegher, Kraut, & Egido 1990), but as the number of agents increases and the agents become geographically and/or conceptually dispersed, supervision of the agents will become increasingly problematic. We argue that agents should be provided with deep domain knowledge that allows them to make justifiable decisions, rather than shallow models of users to mimic. In this paper, we use the application domain of distributed meeting scheduling to investigate how agents embodying deeper domain knowledge can choose among alternative strategies for searching their calendars in order to create flexible schedules within reasonable cost.

Introduction

In characterizing computational tools for supporting cooperative work, researchers have often considered how dimensions of time and space lead to different tools. The result is often a characterization (Johansen 1988) of tools like that shown in Table 1, with representative entries in each of the matrix elements. Implicit in the entries of this matrix is the question being addressed by these tools. To make this explicit, we could say that the matrix categorizes "tools that support collaboration on a task among multiple participants who are in the x", where x is some combination of time and place dimensions.

But consider what happens if we use the same matrix but ask a different, though just as important question. Since most if not all participants in a collaboration are also participating in other collaborations, let us use the matrix to categorize "tools that support collaborations for a **single** participant when the collaborative tasks are in the x".

A person can face tasks at the same place and different times, as visitors (email messages) arrive periodically at the person's office (workstation) in the course of a day. Or the tasks might be at different places at different times, so that the person must travel (perhaps electronically) from task to task during a day. Either way, computational tools such as advanced interfaces and networking software can support the person in his or her collaborations by allowing the person to move more quickly among tasks and to complete each task faster.

Often, though, tasks for various collaborations must be done at the same time. For such tasks, there are two general approaches to providing computational support. One approach is to use computer processes to *prioritize* tasks, which serves to conceptually push the problem back into the "different times" column. These processes can, for example, filter and sort email (Malone *et al.* 1987), and can solve scheduling problems to help the user navigate among competing tasks so as to attend to them one at a time in the proper order. Hence, this approach performs "triage" on the tasks, but the user has to eventually attend to them all.

The other general approach is to *delegate* responsibility for tasks so that they can indeed be handled in parallel. The idea here is to generate, to some degree, processes for the user that act on the user's behalf when he or she is otherwise occupied. Thus, in this approach, the user might never have to attend to some of the tasks. In the case of "same place, same time," these processes could reside at the user-machine interface, intercepting tasks meant for the user and completing them semi-autonomously. Much of the recent work in building "agents" into interfaces has been directed toward this problem. In the case of "different places, same time," the processes could reside (geographically or conceptually) remotely from the user, acting on his or her behalf with little if any supervision on the part of the user. Thus, while interface agents could be monitored and continuously tailored by the user, remote surrogate agents cannot be. A user employing surrogate agents must therefore have confidence in their decision-making, and the agents must have the ability to adapt themselves to changing circumstances based on well-founded criteria. These approaches are summarized in Table 2.

Our work has focused on the problem of how an application domain for intelligent surrogate agents can be analyzed, understood, and represented such that these

	Same Time	Different Times
Same Place	Electronic whiteboards	Electronic bulletinboards
Different Places	Tele-conferencing	Electronic mail

Table 1: Tools for Supporting a Collaboration.

	Same Time	Different Times
Same Place	Filtering agents⟶ Interface agents	Tools for rapid task completion upon task arrival
Different Places	Scheduling agents⟶ Surrogate agents	Tools for rapid movement among distributed tasks

Table 2: Tools for Supporting a User in Multiple Collaborations.

agents can make appropriate adaptations to their environment, to carry out tasks on behalf of human users. Our particular domain of inquiry has been the meeting-scheduling application, and elsewhere we have analytically developed and experimentally verified quantitative predictions of performance for various strategies for proposing, counterproposing, and committing to meetings (Sen & Durfee 1992). In this paper, we consider a different aspect of the problem, namely, strategies for ordering the possible meeting times to propose.

The choice of strategy for considering time intervals for meetings will have numerous effects, including effects on the density of meetings in different parts of the calendar, the likelihood of scheduling future meetings of different types, the costs of scheduling, and the time needed to schedule meetings. More importantly, given targets for calendar densities and limited costs and time for scheduling, a calendar management agent should adapt its strategy choice based on the larger context of what it expects to schedule in the future and what it knows of the calendars of the other agents. These adaptations might not be under the constant supervision of the user, and thus should be made by embedding domain knowledge (a rigorous model of the task) into the agent, rather than trying to capture a superficial model of the user acting in a small sample of cases.

Though we use the domain of meeting scheduling to evaluate the effects of different search biases, our results should hold for any distributed, reactive scheduling systems which respond to resource requests as and when they arrive dynamically over time. The evaluated search biases can also be used in batch mode scheduling systems, but we believe more efficient heuristics are available for such problems (Noronha & Sarma 1991).

Problem Specification

A meeting schedule consists of a group of meetings for a group of persons. Given a set of n meetings and k attendees (hosts and invitees), a scheduling problem is represented as $\mathcal{S} = (\mathcal{A}, \mathcal{M})$, where $\mathcal{A} = \{1, 2, \ldots, k\}$ is the set of attendees and $\mathcal{M} = \{m_1, m_2, \ldots, m_n\}$ is the set of meetings 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 meeting is represented by a tuple:

$$m_i = (A_i, h_i, l_i, w_i, S_i, a_i, \mathcal{T}_i),$$

where

$A_i \subseteq \mathcal{A}$, is a set of attendees of the meeting;

$h_i \in A_i$, is an attendee who will host the meeting;

l_i is the required length of the meeting in hours;

w_i is the weight or priority assigned to the meeting;

S_i gives a set of possible starting times on the calendar for the meeting. If $|S_i| = 1$ the meeting is said to be *constrained* (the exact interval to be used for the meeting, if possible, is pre-specified); if S_i includes all physically possible time slots on the host calendar, assuming that it was empty, that can accommodate a meeting of length l_i starting at that slot, then the meeting is said to be *unconstrained*; otherwise, the meeting is *semi-constrained*;

a_i is the arrival time of the meeting (host gets to know it has to schedule this meeting);

$\mathcal{T}_i$ is the time interval for which the meeting m_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 meeting m_i is scheduled) if the meeting could be scheduled, and by $\emptyset$ otherwise.

The agents use this representation as they engage in a distributed scheduling process based on the multistage negotiation protocol. The protocol involves the following steps. On receipt of a meeting to schedule, the meeting's host searches its calendar for possible time intervals, and proposes the top n ($n \geq 1$) to invitees. An invitee, upon receiving a meeting announcement, will return a bid. The bid can either respond *yes/no* to each of the n proposed times, or it can respond with m possible meeting times, where those times might overlap with the original n but can also counterpropose new *alternative* time intervals. The host, after receiving bids, can attempt to confirm an agreeable time if all of the agents have indicated that a particular time is free for each of them, and otherwise the host will repeat the process with a new announcement message giving a new selection of n meeting times.

Search Bias

We view a search bias as an *a priori* measure of the goodness of certain particular solutions being evaluated. In this respect, different search biases can be related to different *Value Goodness* measures (Haralick

& Elliott 1980), as used in the constraint satisfaction literature.

Our earlier work tacitly assumed that meetings should be scheduled as early as possible, and yielded solutions where calendars tended to be dense at the beginning and sparse at the end. This "lumping" has the disadvantage that it places uneven demands on the user, and that it cannot easily accommodate what we call high-priority short-notice (HPSN) meetings. The latter means that, if the user suddenly learns of a meeting that must be scheduled very soon, there tends to be no open slots for this meeting, leading to costly rounds of cancellation and rescheduling.

More evenly loaded calendars are likely both to place more steady demands on users and to accommodate HPSN meetings more easily. Building such a calendar means that, when scheduling a new meeting, preference should be given to slots that are in the least dense portion of the calendar. Unfortunately, search with this criterion is not nearly as systematic as with the previous (prefer early) criterion, because, while all agents will agree on what is earlier or later, the agents will generally disagree on where the least dense parts of the calendar are, since each has a different calendar with different meetings distributed in it.

Moreover, while the resulting schedule will more likely be able to accommodate HPSN meetings, it will have more trouble scheduling long meetings. That is, scheduling meetings in less dense parts of the schedule tends to fragment available time into smaller and smaller pieces: even if an agent has n hours of free time, those hours might be broken into short spans across several days. Note that a preference for earlier meetings would not have this problem, since it will tend to leave longer contiguous blocks of time at the end of the calendar.

We now present three different search biases, identifying how they work, and the types of solutions (calendar profiles) they generate:

Linear early (LE): With this search bias, an agent attaches increasing goodness values to intervals earlier in the calendar, i.e., given a meeting, the agents try to schedule the meeting as early as possible. The search bias is implemented by making an agent start searching the calendar at the earliest possible scheduling opportunity, skipping over any intervals overlapping with already scheduled meetings, and negotiating with the earliest free interval on the calendar long enough to accommodate the meeting. This search bias can be likened to the *First Fit* memory allocation scheme (Peterson & Silberschatz 1985).

Linear least dense (LLD): With this search bias, an agent tries to schedule a meeting in the least dense part of its calendar. The search bias is implemented by the host of the meeting ranking all the empty intervals on its calendar long enough to accommodate a given meeting (and within the window of acceptable times for the meeting) by a function that measures the number of free calendar slots around that interval. The agent then steps down this ranked list and negotiates with other attendees of the meeting until it can schedule the meeting. This search bias can be likened to a local form of *Worst Fit* memory allocation strategy (Peterson & Silberschatz 1985).

Hierarchical (H): With this search bias, an agent tries to schedule a meeting in the least dense part of the combined search space of all the attendees of the calendar. The search bias is implemented by building an abstraction hierarchy atop the linear calendar for each meeting-scheduling agent. At each node in the hierarchy, agents keep a record of the number of intervals of different length free below that node in the hierarchy. The calendar space lends itself to a very natural hierarchy of hours, days, weeks, etc., and the agents participating in a meeting can first identify a good week to meet in, then identify a good day within that week, and finally an actual interval within that day. Given a meeting of some particular length to schedule, the host asks for and receives information from all the invitees about how many intervals of that length are open at each node (e.g., at each week) of the highest level of the hierarchy. It multiplies the numbers together for corresponding nodes, ranks the nodes, elaborates the best one, and proceeds to repeat the process for the next level of the hierarchy under the elaborated node. At the ground level, information exchange takes place like the **LE** scheme. Backtracking occurs if a particular portion of the ground level being elaborated contains no solution to the scheduling problem. In this paper, the levels of the hierarchies used by the agents correspond to days and hours only. Agents first negotiate about the likelihood of scheduling a meeting on different days, then choose the most likely day and start **LE** negotiation within that day. The host may fail to schedule the meeting in the most likely day, and will then backtrack to the next most likely day. This search bias can be likened to a global form of *Worst Fit* memory allocation strategy (in the sense that the fitting takes place in the most free part of the combined search space of all the participants of a meeting).

Experimental Results and Analysis

We evaluated these search biases both in terms of achieving acceptable solutions and in terms of computational and communication costs using the following metrics:

Communication cost (CC) is measured by the average number of information packets exchanged to schedule a multi-agent meeting. One information packet consists of a proposal from a host or an invitee. In the case of **H** search bias, while negotiating at a non-leaf node of the hierarchy, a node-number and meeting-likelihood pair is considered an information packet. This measure provides us with an estimate of the amount of bandwidth required to schedule meetings.

$$C = \frac{\sum_{i=1}^{n} \sum_{j=1}^{k} c_{ij}}{n_m}$$

where $n = |\mathcal{M}|$, $k = |\mathcal{A}|$, and $c_{ij} = \sum_{\forall y, y \in A_i \text{ and } j \neq y} |\Pi_{ijy}|$, where Π_{ixy} denotes the set of proposals that individual x sent to individual y to schedule a meeting m_i, and n_m is the number of multi-agent meetings scheduled. This measure is used to distinguish otherwise equivalent biases by preferring those which requires less communication bandwidth.

Iterations (I) required to schedule a meeting is measured by the average number of rounds of negotiation entered into by the participating agents in a meeting before a meeting is scheduled, or it is recognized that the meeting cannot be scheduled. This measure indicates the amount of time required to schedule a multi-agent meeting.

Slots Searched (SS) to schedule a meeting is measured by calculating the sum of the number of possible intervals on the calendar looked at by the participants while trying to schedule a meeting. The measure is an average over all the meetings scheduled. It is indicative of the search complexity (and correspondingly, the time taken) for finding intervals to propose.

Meeting Hours Missed (MHM) represents the success of scheduling the meetings requested so far, and is calculated as the ratio of the number of hours that got scheduled over the number of hours for all meetings requested.

$$\text{MHM} = \frac{\sum_{i=1}^{n} l_i * |A_i| * (1 - \rho_i)}{|\mathcal{A}|},$$

$$\text{where } \rho_i = \begin{cases} 1 & \text{if } m_i \text{ has been scheduled} \\ 0 & \text{otherwise.} \end{cases}$$

Density Profile Characteristics (DPC) are plots that display the variation of the number of free intervals of different length over the length of the calendar. The numbers are averaged over all the agent calendars. This measure is indicative of the spread of the meetings scheduled on the calendar, and can be used to predict the success of scheduling different kinds of meetings on the given calendar.

The first three of these criteria measure the cost incurred in the process of scheduling the meeting, whereas the last two criteria reflect the quality of the schedule generated both in terms of how well it has scheduled known meetings and how likely it is to schedule future meetings. We now briefly present our expectations of how different search biases will perform on the different evaluation criteria, and also provide brief intuitions behind these expectations.

We report results from two different sets of experiments involving two different organizations, where an organization is characterized by the number of individuals scheduling meetings and the length of their corresponding calendars. For the smaller organization, we consider 3 agents with 5 day calendars. The corresponding numbers for the larger organization are 10 and 14 respectively. In both cases, each day consists of 9 hours. The agents start with an empty calendar and are given a number of meetings to schedule. The meeting requests are assigned such that if all the meeting requests get scheduled, each of the agents will have $\mathcal{H}$ hours reserved in their calendar. Given a total number of L =45 (126) hours on each agent's calendar for the smaller (larger) organization, we have run experiments varying $\mathcal{H}$ from 10 up to 40 (70 up to 130), in steps of 5 (10). All meetings are totally unconstrained. For each search bias, and for each value of $\mathcal{H}$, the results reported are averaged over 1000 runs of randomly generated meetings. The meeting lengths are chosen from a discrete probability distribution assigning the probabilities 0.3, 0.25, 0.2, 0.15, 0.05, 0.05, 0.0, 0.0, and 0.0 to meeting lengths 1, 2, 3, 4, 5, 6, 7, 8, and 9 hours respectively. In the simulation, we schedule meetings sequentially. Though one of the primary benefits of a distributed formulation of the scheduling problem is the increased throughput obtained by concurrent processing of multiple tasks, we use a sequential mode here to identify the effects of search bias independent of concurrent processing issues.

For the smaller organization, the number of iterations and the communication cost obtained with the **LE** bias is low and varies little with varying $\mathcal{H}$. Communication cost is roughly 3 times the number of iterations, because multi-agent meetings can involve either 2 or 3 persons, and the corresponding communication cost per iteration of the scheduling process is 2 and 4 respectively (more generally, for a k person meeting, the communication cost per iteration of the normal information exchange phase of our negotiation protocol is $2(k-1)$). Low values of these metrics can be attributed partly to the use of an effective **bidding** strategy (*alternatives* option). The fact that the **LLD** bias performed only slightly worse on these metrics indicates that other reasons contributing to these results include the relatively empty calendars available to the agents to process meeting request, the small number of agents, and small number and high percentage of multi-agent meetings (which means that the calendars fill up similarly). This analysis prompted us to experiment with larger organizations to see how these biases scaled up.

Results from experiments with the larger organization are presented in Figure 1 and Figure 2. In these experiments, 75% of the meetings required the attendance of two or more agents.

In sharp contrast to the results obtained with the smaller organization, in experiments conducted with the larger organization, the **H** search bias schedules meetings quicker than the **LE** or **LLD** search biases. This is due to the fact that when attendee calendars are stacked up differently, the **H** bias use the hierarchical information exchange to quickly identify parts of the calendar that are likely to contain a mutually acceptable interval. The other search biases waste more time eliminating unlikely candidates from the search process. This is particularly true for the **LLD** search bias, in which case the host does not receive informative replies from the invitees to aid the negotiation process. The number of iterations taken to schedule meetings with the **H** search bias is almost constant over $\mathcal{H}$. One of these iterations is spent to gather information about the density profiles at the day level of the hierarchy from the invitees of a multi-agent meeting. This particular iteration also incurs a heavy communication cost as the number of information packets sent by each invitee is equal to the number of days in the calendar, as opposed to sending only one packet for any other iteration. As such, the communication cost is considerably higher than in the other two cases.

The slots searched metric for the **LE** bias linearly increases with $\mathcal{H}$; this was anticipated because with more hours full on the calendar, the agent has to search progressively further to find an empty interval on the calendar. As expected, the slots searched by the **LLD** bias is extremely large: all possible intervals are looked at to determine if they can accommodate the given meeting. The number of slots searched is extremely small as the information maintained in the hierarchy helps in quickly identifying potential intervals to be used for a meeting. The values for this metric grows slightly with increasing $\mathcal{H}$ to reflect increasing search at the ground level by an invitee to come up with an *alternative* bid to a proposal from the host, as more and more hours on the calendar are reserved for other meetings.

The observed metric of meeting hours missed with different search biases is as expected, since agents start failing to schedule meetings only when $\mathcal{H}$ is close to L. Most of the meetings missed are either multiple agent meetings or are long meeting requests arriving at the end of the scheduling run. The value for this metric is noticeably greater for the **LLD** and **H** bias than that for **LE** bias for $\mathcal{H}$ ranging from 80 to 120. In its attempt to evenly schedule meetings across the calendar length, the **LLD** search bias ends up fragmenting the calendar space, and hence is unable to accommodate a percentage of long meetings as $\mathcal{H}$ increases above a threshold value. An interesting result of our experiments with the two organizations was that, though the **LE** search bias leads to smaller values of meeting hours missed in the larger organization, the proportional savings obtained are far less than in the smaller organization. This is because even though each agent is relatively free towards the latter part of their schedules when using the **LE** bias, individual schedules can differ significantly in the exact intervals that are free in these portions. As such, even with space compaction, short meetings with large numbers of attendees may not get scheduled.

Figure 2 shows that when using the **LE** bias, more and more intervals of any given length are open on the calendar as one proceeds from the front to the end of the calendar. On the other hand, the **LLD** and **H** search biases are able to deliver on the promise of even density profiles across the length of the calendar. Results are better than expected for the **LLD** bias because this bias only balances the host's calendar, but that seems sufficient in balancing each agents' calendar since everyone gets to be the host with roughly equal frequency, and more importantly, they have a number of meetings with themselves which smooth out their respective density profiles.

Observations

There are several important observations to be made given the results and analysis of the last section; some of these reinforce our expectations, while a few provide new insights to the properties of the search biases. Observations generally true for all the search biases considered are:

1. for any given $\mathcal{H}$, the number of open intervals increases with decreasing length of the intervals (in a free interval that can accommodate a long meeting, we can schedule a number of smaller meetings),
2. the number of intervals open for a given length is a non-increasing function of $\mathcal{H}$ (this is observed by comparing different graphs like Figure 2 with varying $\mathcal{H}$).

Both **LLD** and **H** search biases generate even density profile characteristics (**DPC**) across calendar lengths in contrast to the skewed density profiles produced by the **LE** search bias. On the other hand, the **LE** search bias leaves room for scheduling long meetings, and hence, particularly in cases where $\mathcal{H}$ is close to L, results in fewer meeting hours missed (smaller **MHM** values) than the **LLD** and **H** search biases. The purpose of building evenly dense profiles is to retain the flexibility of scheduling HPSN meetings. If that flexibility is obtained at the price of being unable to schedule some regular, run-of-the-mill meetings, then that is often too high a price to pay. So, depending on the importance of HPSN meetings, one may either want to use the **LLD** or **H** search biases in favor of the **LE** search bias across the range of $\mathcal{H}$ values, or to $\mathcal{H}$ values below a threshold only.

Though meetings can be quickly scheduled (smaller **I**) in small organizations by using the **LE** search bias, as we consider larger organizations, it is clear that the **H** bias is the most expedient bias for scheduling meetings. Communication cost required to schedule meet-

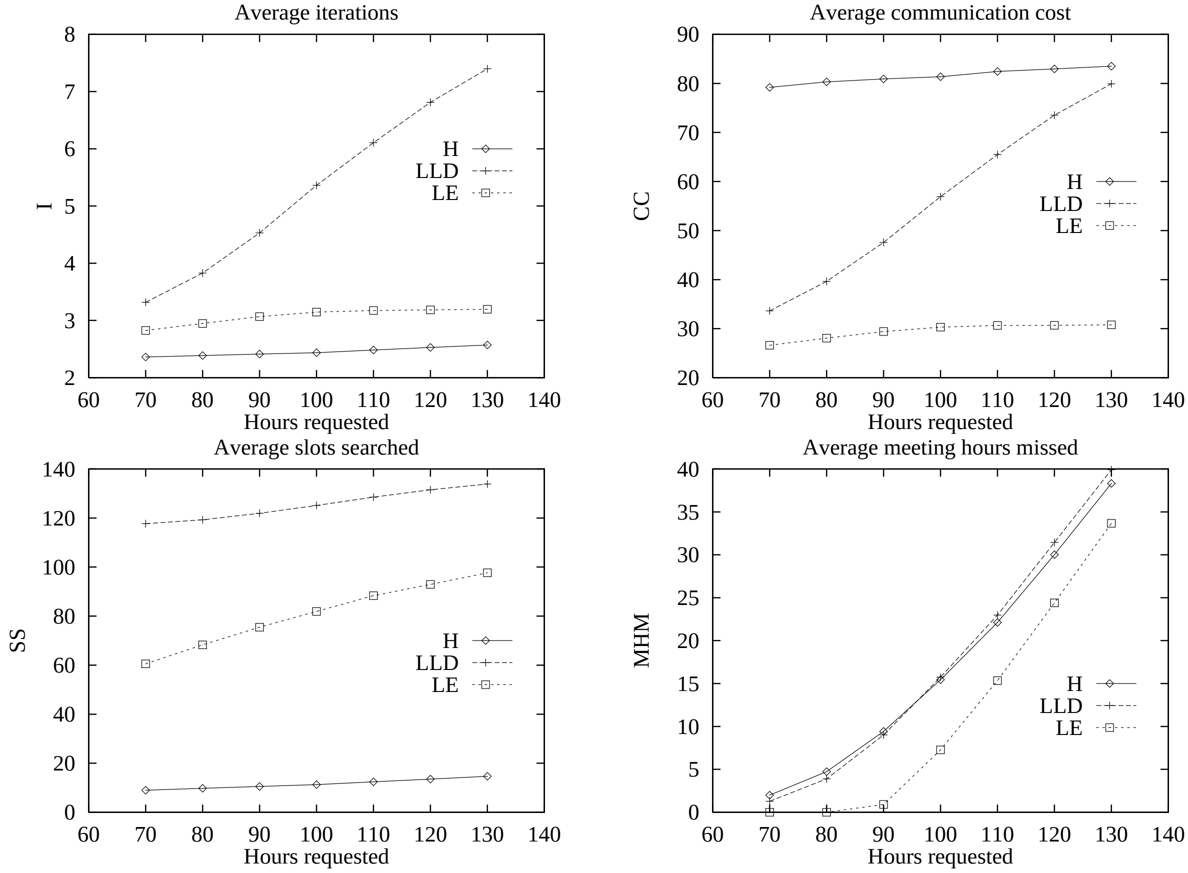

Figure 1: Performance of the **LE**, **LLD**, and **H** search biases on the metrics evaluating the scheduling cost (**MHM**,**CC**, **I**, and **SS**) as $\mathcal{H}$ varies from 10 to 45.

ings is greater for **H** search bias than the other two (of which, the values for **LLD** are higher than those for **LE** search bias), because of the increased overhead of the first iteration (as information about more days needs to be exchanged). This problem may be solved by multiple levels of the hierarchy; if 10 days were separated into two weeks, and a solution was found in the least dense of the two weeks, communication cost for the 5 days in the other week could be saved.

In terms of the slots searched metric, the **H** search bias is a sure winner. In our past work, we have noted that, in situations where concurrent scheduling of several meetings is taking place, greater time spent in local search will lead to increased probability of mutually harmful interactions between scheduling processes that share the same calendar (Sen & Durfee 1994). So, although the significance of this metric is not apparent in this simulation (with no concurrent scheduling), in real-life this metric will greatly impact the other cost metrics.

For all the search biases considered, we notice a decrease in scheduling efficiency (as measured by the meeting hours missed metric) with an increase in the size of the organization. However, in larger organizations (with more agents, longer calendars), the **H** search bias takes the least number of iterations, provides more flexibility, and does not significantly sacrifice success rate of ordinary (non-HPSN) meetings (as compared to the **LE** search bias). Hence, we prescribe a switch in search bias from **LE** to **H** as the organization grows. The use of **LLD** should be limited to situations where, even though the organization is small, HPSN meetings are fairly common.

More detailed analysis of these runs revealed that given the same calendar states of attendees, different meetings (as characterized by number of attendees, their calendars, meeting length, etc.) could be more efficiently scheduled by different search biases. More importantly, the density profiles produced by one search bias led to calendar states on which meetings can be scheduled more efficiently by another search bias (for example, scheduling early produces front-loaded calendar, on which it is easy for the **H** and **LLD** biases to locate the least dense portions). So, an autonomous scheduler can perform better by switching between the different search strategies as and when appropriate. In the following we present some analytical results that enables these agents to decide on the most appropriate search bias when asked to schedule a meeting.

Adaptive scheduling

The above experimental evaluation allows us to characterize the relative strengths and weaknesses of the proposed search biases. To improve the efficiency of the meeting scheduling agents, we need to derive precise analytical expectations of the performance, as measured by some given performance metrics, of these search biases. This will allow the implementation of intelligent, autonomous scheduling agents capable of adapting to changing environmental conditions. In the following, we present an analysis of the number of iterations required to schedule meetings with the **LE** and **H** search biases.

We first consider the **LE** search bias with the invitee agents responding with acceptance or rejection messages (no counterproposals). In this scenario, a host agent is trying to schedule a meeting with A invitees. Let $K = \mathcal{L} - l + 1$ be the number of places in which a meeting of length l can a start on a working day of length $\mathcal{L}$. For invitee x, let $n_x \leq K$ be the number of these intervals open on the day under question. If the host was using the one proposal per iteration, the probability that it will take i iterations to schedule the meeting is given by (Sen 1993)

$$P_{i,1} = \sum_{j=0}^{i-1}(-1)^j \begin{pmatrix} i-1 \\ j \end{pmatrix} \frac{\prod_{x=1}^{A} \begin{pmatrix} K-j-1 \\ n_x - j - 1 \end{pmatrix}}{\prod_{x=1}^{A} \begin{pmatrix} K \\ n_x \end{pmatrix}}. \tag{1}$$

If the host was proposing N intervals per iteration, the probability that it will take i iterations to schedule the meeting is given by

$$P_{i,N} = \sum_{j=N(i-1)+1}^{N*i} P_{j,1} \tag{2}$$

Now we analyze the **H** search bias. Let us assume P agents (numbered $1, \ldots, P$) are involved in scheduling a meeting of length l, and they have constructed identical temporal abstraction hierarchies over the base calendar space (linear ordering of calendar hours). For any internal node in the hierarchy, we will calculate the probability that one or more common intervals are free in the base space of the calendars under that node, for every attendee of the meeting. Let x be the node in question. Since the hierarchies formed by the agents are identical, every agent has $a(l,x)$ intervals of length l below this node of its abstraction hierarchy. Let the number of intervals of length l currently free under node x for agent i be $f_i(l,x)$. We can then calculate the following (see (Sen 1993) for details of the calculation):

$$P\{\text{At least one interval is free in each attendees calendar under node } x\}$$
$$= \sum_{j=1}^{a(l,x)}(-1)^{j+1} \begin{pmatrix} a(l,x) \\ j \end{pmatrix} \frac{\prod_{i=1}^{P} \begin{pmatrix} a(l,x)-j \\ f_i(l,x)-j \end{pmatrix}}{\prod_{i=1}^{P} \begin{pmatrix} a(l,x) \\ f_i(l,x) \end{pmatrix}}. \tag{3}$$

Given the two level hierarchy (days and hours) that we are considering in this paper, probabilities for scheduling under each day is calculated using Equation 3. The days are then sorted by these probabilities, and the agents negotiate over days in decreasing order of this probability until a mutually free interval is found.

Given a particular scheduling situation, we can develop probability mass functions and probability distribution functions of the random variable corresponding to the iteration at which scheduling is completed (using Equations 1 or 2 for the **LE** search bias and Equation 3 for the **H** search bias). This information can be used to calculate the expected number of trials to schedule the meeting using the two announcement strategies. The strategy that leads to a lower expected number of trials to schedule the meeting is then adopted. This method of search bias selection is completely flexible in the sense that given any new meeting to schedule, the most appropriate choice is made. We plan to experimentally evaluate the savings obtained by such a flexible choice of bias against the cost of fixed bias scheduling.

Conclusion

In this paper, we have argued in general terms for developing surrogate agents that make decisions based on carefully-constructed models of the application domain, and we have more specifically highlighted the importance of retaining flexibility for future resource requests when searching for a solution to a current resource scheduling problem. We have implemented and evaluated several approaches for biasing the distributed search process in a distributed meeting scheduling application. Our results indicate that hierarchical distributed search will be increasingly important as the scheduling problems are scaled up to many agents and long schedules. More importantly, we have identified the need for an adaptive choice of search bias, and provided a probabilistic model that allows us to make this choice appropriately.

Our future efforts will involve addressing broader class of scheduling problems using the techniques developed here. We are particularly interested in the dynamic, concurrent scheduling of a multitude of resource-constrained project networks (possibly managed by different departments in an organization) with significant resource interdependencies (Bell 1989). We also believe that our approach of distributed, incremental scheduling in a dynamic domain can be successfully applied to a wide variety of scheduling problems on which other AI techniques have been used (Noronha & Sarma 1991). In particular, we have showed that our proposed system of distributed contract-based negotiation can be effectively used in a manufacturing environment (Sen & Durfee 1993). We are currently implementing an distributed meeting scheduler on a local area network supporting workstations, PCs, and Macintoshes.

Acknowledgements

This research has been sponsored, in part, by the National Science Foundation under Coordination Theory and Collaboration Technology grant IRI-9015423 and a Research Initiation Award IRI-9410180, and by a grant from Bellcore.

References

Bell, C. E. 1989. Maintaining project networks in automated artificial intelligence planning. *Management Science* 35(10):1192–1214.

Galegher, J.; Kraut, R. E.; and Egido, C. 1990. *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Haralick, R., and Elliott, G. 1980. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence* 14(3):263–313.

Johansen, R. 1988. *Groupware: Computer Support for Business Teams.* New York, NY: Free Press.

Malone, T. W.; Grant, K. R.; Turbak, F. A.; Brobst, S. A.; and Cohen, M. D. 1987. Intelligent information-sharing systems. *Communications of the ACM* 30(5):390–402.

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

Peterson, J. L., and Silberschatz, A. 1985. *Operating System Concepts.* Reading, MA: Addison Wesley.

Sen, S., and Durfee, E. H. 1992. A formal analysis of communication and commitment in distributed meeting scheduling. In *Working Papers of the 11th International Workshop on Distributed Artificial Intelligence*, 333–342.

Sen, S., and Durfee, E. H. 1993. Dependent subtask processing in a contract-net for manufacturing. In *Proc. of AAAI-93 Workshop on Intelligent Manufacturing Technology*, 7–13.

Sen, S., and Durfee, E. H. 1994. The role of commitment in cooperative negotiation. *International Journal of Intelligent and Cooperative Information Systems* 3(1):67–81.

Sen, S. 1993. *Predicting Tradeoffs in Contract-Based Distributed Scheduling.* Ph.D. Dissertation, University of Michigan.

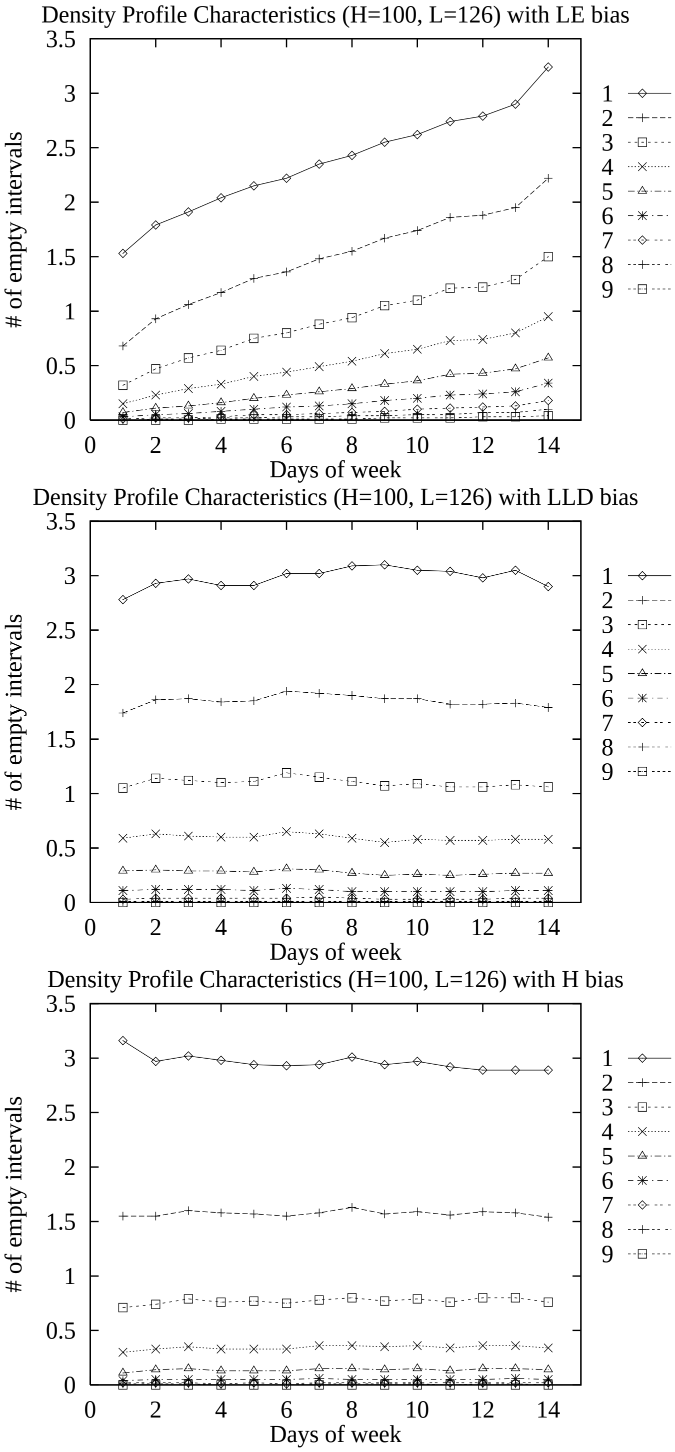

Figure 2: Performance of the **LE**, **LLD**, and **H** search biases (from left to right) on the metric evaluating quality of schedule generated (**DPC**) for the case where $\mathcal{H} = 100$.