Bumping Strategies for the Multiagent Agreement Problem

Pragnesh Jay Modi and Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
{pmodi,veloso}@cs.cmu.edu

ABSTRACT

We introduce the Multiagent Agreement Problem (MAP) to represent a class of multiagent scheduling problems. MAP is based on the Distributed Constraint Reasoning (DCR) paradigm and requires agents to choose values for variables to satisfy not only their own constraints, but also equality constraints with other agents. The goal is to represent problems in which agents must agree on scheduling decisions, for example, to agree on the start time of a meeting. We investigate a challenging class of MAP – private, incremental MAP (piMAP) in which agents do incremental scheduling of activities and there exist privacy restrictions on information exchange. We investigate a range of strategies for piMAP, called "bumping" strategies. We empirically evaluate these strategies in the domain of calendar management where a personal assistant agent must schedule meetings on behalf of its human user. Our results show that bumping decisions based on scheduling difficulty models of other agents can significantly improve performance over simpler bumping strategies.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms

Algorithms

Keywords

Distributed Constraint Optimization

1. INTRODUCTION

Distributed Constraint Reasoning (DCR) [2, 11, 12, 18, 20] has been proposed as a theoretical foundation for problems in multiagent systems, for example, distributed scheduling problems. In DCR, a set of variables are distributed among a set of agents and constraints among variables require agents to coordinate their value choices. This paper considers the DCR approach for problems where an agent must schedule activities under the following three conditions:

- Schedules are inter-dependent with other agents. A schedule is *valid* if it satisfies both local constraints and external constraints with other agents. For example in meeting scheduling, a person has local constraints such as "attend one meeting at a time" and also has external constraints like "other attendees must agree on the time of a meeting".
- Schedules are built incrementally. That is, new activities must be incorporated into an existing valid schedule to produce a new valid schedule. A key feature of incremental scheduling is that existing activities often need to be moved, or "bumped" and rescheduled, in order to successfully accommodate the new activities.
- Schedules contain private information and each agent retains ownership of its schedule. We assume this as an explicit property of the application domain. This property eliminates a solution approach in which all information is communicated to a central scheduler that constructs a global schedule for all agents. Instead, each agent makes its own scheduling decisions and communicates with others to ensure a valid schedule. Importantly, the assumption of private information places limits on the information that is exchanged.

We argue that the above are key essential features of many real-world distributed scheduling problems. Incremental scheduling is clearly an important class of problem. Inter-dependencies as defined by equality constraints arise whenever multiple agents must schedule a joint activity that must be executed at the same time, e.g., scheduling a coordinated invasion in military mission planning. Finally, privacy restrictions are ubiquitous when agents are used to represent the interests of humans.

There is currently a mismatch between existing approaches to DCR and what is needed to solve scheduling problems with the above features. First, existing DCR representations such as DisCSP [20] and DCOP [12], which make few assumptions about constraints, are overly general for some domains. Algorithms designed for these representations fail to exploit additional assumptions that may be available. For example, if only equality constraints are needed to encode the underlying problem, it makes sense to limit the representation and design a more specialized (and potentially more efficient) algorithm. Currently, such specialized representations are lacking.

Second, existing work in DCR has not explicitly considered incremental problem solving and rescheduling (bumping) as a key algorithmic decision point. Instead, most existing approaches, with the exception of the Open CSP approach of Faltings and Gonzalez[7], have focused on batch problem solving. A batch approach is potentially inefficient in an incremental setting because it fails to take advantage of the existing solution.

Finally, previous research in classical scheduling has shown that using heuristics can significantly aid in problem solving [1], but most existing methods in DCR, e.g., DBO[20], AWC[20], Adopt[12], use uninformed search. Effective search heuristics for DCR have the potential to improve performance. For example, general high-level information about other agents, such as the number of variables they own, could be incorporated as a heuristic. The most progress to date on this idea is the texture measures approach of Sycara et. al. [19], discussed further in Section 5.

This paper presents a DCR approach to the multiagent scheduling problem and the issues described above. First, to address the issue of lack of specialized representations and lack of focus on incremental problem solving, we introduce the private, incremental Multiagent Agreement Problem (piMAP) as a special class of DCR in which constraints between agents are limited to equality constraints. Agents assign values to variables in an incremental manner with restrictions on to whom certain information may be communicated. Specifically, piMAP defines a set of participants for each variable and explicitly prohibits the communication of information about variables between agents who are not participants in the variable.

Second, in order to address the issue of lack of use of heuristics in DCR, we investigate a range of heuristic strategies for the "bumping" problem in piMAP. The bumping problem is deciding how to rearrange the existing schedule in order to schedule a new activity. We introduce a heuristic in which the main idea is to represent and exploit general knowledge about other agents (while adhering to privacy constraints) and their "scheduling difficulty". Our notion of scheduling difficulty is sufficiently general to model a wide range of contributing factors. The specific scheduling difficulty model we investigate in this paper assumes an agent has or can obtain knowledge of the average schedule density of other agents.

Finally, we use piMAP to model the multiagent meeting scheduling problem and investigate the performance of our bumping strategies in this domain. Multiagent meeting scheduling has been investigated before [9, 17, 8] but our work is distinctive in its focus on bumping techniques for rescheduling in an incremental setting. Existing work has not investigated this aspect of the multiagent meeting scheduling problem. We evaluate our approach in an experimental testbed for multiagent meeting scheduling where personal assistant agents schedule meetings on behalf of their human users. We simulate a human organization where higher ranked people have very busy calendars and lower ranked people have lower calendar density. Our results show a significant reduction in scheduling failure rate with a bumping strategy that uses a model of scheduling difficulty against other strategies that do not.

2. PROBLEM DEFINITION: PIMAP

A Distributed Constraint Reasoning (DCR) problem is defined by a set of agents, variables, values and constraints, where each variable is assigned to an agent who has control of its value. Constraints which are local, i.e., among variables assigned to the same agent, are called *intra-agent* constraints, while constraints which are external, i.e., among variables assigned to different agents, are called *inter-agent* constraints. DCR can be viewed as a distributed form of the well-known and very successful CSP representation from AI [4].

We use the general DCR framework to define the *multiagent agreement problem* (MAP). MAP is a special class of DisCSP as proposed by Yokoo and others [20]. The key differences are discussed after the following formal definition.

2.1 Formal Definitions

In MAP, a set of agents must map elements from one set, which are modeled as the variables, to elements of a second set, which are modeled as the values. Importantly, inter-agent constraints require multiple agents to agree on the assignment of a value to a shared variable.

We define the *multiagent agreement problem* (MAP) as follows:

- $\mathcal{A} = \{A_1, A_2, ..., A_n\}$ is a set of *agents*.
- $\mathcal{V} = \{V_1, V_2, ..., V_m\}$ is a set of *variables*.
- $\mathcal{D} = \{d_1, d_2, ..., d_k\}$ is a set of *values*. Each value can be assigned to any variable.
- *participants*$(V_i) \subseteq \mathcal{A}$ is the set of agents assigned the variable V_i. A variable assigned to an agent means it has (possibly shared) responsibility for choosing its value.
- *vars*$(A_i) \subseteq \mathcal{V}$ is the set of variables assigned to agent A_i.
- For each agent A_i, C_i is an *intra-agent* constraint that evaluates to true or false. It must be defined *only* over the variables in *vars*(A_i).
- For each variable V_i, an *inter-agent* "agreement" constraint is satisfied if and only if the same value from $\mathcal{D}$ is assigned to V_i by all the agents in *participants*(V_i).

We say an assignment of values to variables is *valid (sound)* if it satisfies both inter-agent and intra-agent constraints. We say an assignment is *complete* if every variable in $\mathcal{V}$ is assigned some value. The goal is to find a valid and complete assignment.

There are two key differences between MAP and DisCSP. First, MAP allows a variable to be shared among a set of agents (participants) while DisCSP assigns each variable to a unique agent. However, MAP can be viewed as a DisCSP by giving a copy of each shared MAP variable to each participant and adding inter-agent equality constraints between the copies. Any DisCSP cannot be converted to a MAP because DisCSP admits general inter-agent constraints, but MAP inter-agent constraints are limited to the equality constraints on shared variables.

A motivation for introducing the MAP representation with shared variables is to conveniently and explicitly capture problems where multiple agents are involved in a joint decision. This is a feature of many distributed domains where each agent brings its own private constraints to bear on the decision, but yet agents must come to an agreement. Another important motivation is to develop more specialized DCR algorithms and approaches that are tailored to this particular problem rather than exclusive focus on using the most general DCR representation and algorithms.

The second key difference from general DisCSP is that MAP assumes a single set of values $\mathcal{D}$, i.e, all variables have the same domain. But this is not a restriction. Given a set of variables with different domains, we can define a new universal domain as the union of the individual domains and add unary constraints to each variable to eliminate infeasible values.

Private, Incremental MAP (piMAP) is an extension to MAP in which agents must solve MAP in an incremental fashion while limiting the information they can exchange:

- **Incremental:** In incremental MAP, new variables and associated constraints are added to the problem over time and must be integrated into an existing assignment. In meeting scheduling for example, new meetings arise over time and must be scheduled in the context of an existing schedule. Given a MAP with agents $\mathcal{A}$ and variables $\mathcal{V}$, an incremental MAP also includes:
  - $S_{init} = \{(V_1, d_1), (V_2, d_2), ..., (V_m, d_m)\}$ is an initial assignment of values to variables in $\mathcal{V}$.
  - $\mathcal{V}' = \mathcal{V} \cup \{V_{m+1}\}$ is a set of variables to be assigned a value.
  - *participants*(V_{m+1}) is a set of agents who are assigned the variable V_{m+1}.
  - $\mathcal{A}' = \mathcal{A} \cup$ *participants*(V_{m+1}) is a set of agents.

  The goal is find a valid and complete assignment for the variables in $\mathcal{V}'$. This incremental aspect of the problem raises the need for the bumping strategies described in this paper. Given a final solution S_{final}, we say a variable $V_i \in \mathcal{V}$ with initial value $(V_i, d_i) \in S_{init}$ was *bumped* if $(V_i, d_i) \notin S_{final}$. That is, V_i is assigned a final value different from its initial value or in the case of an incomplete solution, unassigned a final value. The total number of bumped variables measures the amount of schedule disruption that is needed to schedule the new variable. All other things equal, an algorithm that is able to obtain a solution with fewer bumps is more desirable than one that requires greater bumps.

- **Privacy:** The information that may be exchanged among agents is limited due to a desire to maintain distribution and privacy. In particular, we assume the following condition.
  - $A_i \in$ *participants*(V_j) does not communicate information about V_j to any agent who is not in *participants*(V_j)

  For example, a variable's current value or the participants of a variable are not communicated to any agent who is not a participant in the variable.

  Finding a solution to MAP under this condition is challenging in part because the indirect constraints that arise through chains of constraints often cross privacy boundries and so cannot be made easily visible to any single agent.

2.2 Meeting Scheduling as piMAP

Multiagent meeting scheduling requires a set of agents $\mathcal{A} = \{A_1, A_2, ..., A_n\}$ to pair a set of meetings $\mathcal{M} = \{M_1, M_2, ..., M_m\}$ with a set of timeslots $\mathcal{T} = \{T_1, T_2, ..., T_p\}$ according to a set of constraints. For simplicity, we assume each meeting has the same duration d, and $\mathcal{T}$ is a set of discrete non-overlapping timeslots of length d. A valid solution must satisfy three constraints: a) each meeting is assigned to exactly one timeslot, b) no attendee is required to attend more than one meeting at the same time, and c) all the attendees of a given meeting agree on its assigned timeslot. We represent this problem using piMAP as follows.

We define a piMAP variable V_i for each meeting M_i, and an piMAP value d_j for each timeslot T_j. The *participants* of variable V_i correspond to the attendees of meeting M_i. The piMAP inter-agent agreement constraint ensures that meeting attendees agree on the start time of the meeting. The piMAP intra-agent constraint C_i is satisfied if and only if no value from $\mathcal{T}$ is assigned to more than one variable in *vars*(A_i), i.e., no timeslot is double-booked in an agents schedule. Although beyond the scope of this paper, C_i could also be used to represent time-of-day preferences (e.g., "no meetings before 11am") or more complex local constraints such as travel time between meetings or back-to-back preferences [15].

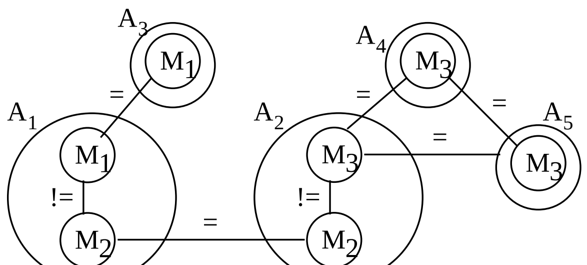

Figure 1: Meeting Scheduling as the Multiagent Agreement Problem.

Figure 1 illustrates the multiagent agreement problem with five agents A_1, A_2, A_3,A_4,A_5 and three meetings M_1,M_2,M_3. Participants are defined as *participants*$(M_1) = \{A_1, A_3\}$, *participants*$(M_2) = \{A_1, A_2\}$, *participants*$(M_3) = \{A_2, A_4, A_5\}$. Variables within an agent must have different values corresponding to different start times, while the variables corresponding to the same meeting must be assigned the same value to satisfy the inter-agent agreement constraint.

3. SOLUTIONS FOR PIMAP

We describe a protocol for piMAP that guarantees a valid assignment. The protocol does not guarantee a complete assignment because additional complexity would be necessary. Instead of focusing on complexity of the protocol, our main purpose is to use this protocol to support investigation of bumping strategies.

3.1 Iterative Agreement Protocol

The Iterative Agreement Protocol (IAP) described in this section is used to obtain valid solutions to a piMAP problem. It is similar to the protocol outlined by Sen and Durfee [17]. Each variable V_i has a unique participant who is the designated *initiator* of V_i. The initiator proposes a single value and collects responses from the other participants in a sequence of *rounds*. In each round, the initiator sends a single proposal (V_i, d_i) and each participant decides whether to accept or reject the proposal. Each participant follows these steps:

- If the proposed assignment does not violate the participant's intra-agent constraint C_i, it accepts the proposal immediately.
- Else, the agent uses a bumping strategy to determine whether to accept or reject the proposal. (We will discuss bumping strategies in the remainder of this paper.)
- If the participant accepts the proposal, it tentatively reserves d_i for V_i and will reject any future proposals that conflict with this tentative assignment.

The initiator collects the responses from all participants in each round and follows these steps:

- The initiator checks if all participants have accepted the current round's proposal.
- If yes, the assignment is confirmed with all participants in one additional round of messages and everyone releases all other tentatively reserved values for V_i if any.

- If no, the protocol continues in rounds until the initiator has no more values to propose, in which case the initiator declares failure and all agents release their reserved values.

3.2 Bumping Strategies

In the Iterative Agreement Protocol, when agent A_i receives a value assignment proposal (V_i, d_i), it must choose whether to accept or reject the value d_i. A bumping strategy is a rule employed by an agent to make this decision. The is a key algorithmic decision point that can have a large effect on amout of schedule disruption and scheduling failure rate.

Possible strategies differ in the amount of knowledge that is assumed. We introduce a set of strategies that range from completely uninformed to increasing amounts of knowledge. In the informed strategies, the idea is to use knowledge about agents to predict which variables will be difficult to reschedule and then avoid bumping them.

3.2.1 Uninformed

These simple fixed strategies require no knowledge.

- **Always Strategy:** Always accept a proposal, and reschedule bumped variables to resolve conflicts.

- **Never Strategy:** Never accept a proposal if the proposed assignment results in a conflict.

3.2.2 Simple Informed

This strategy requires knowledge of the number of participants of variables, but requires no further knowledge of the other agents.

- **NumParticipants Strategy:** If a proposal (V_i, d_i) conflicts with current assignment (V_j, d_j), accept the proposal only if the size of *participants*(V_j) is less than the size of *participants*(V_i).

The intuition is that variables with fewer participants are easier to reschedule and so should be bumped in favor of variables with greater participants.

What is the maximum number of bumps possible when agents use this strategy to schedule a variable V_i with n participants? In the worst case, the initiator proposes to the other $n-1$ participants a value that conflicts with a unique variable in each's local schedule and each such variable has $n-1$ participants. Since $n-1 < n$, each of the $n-1$ participants of V_i will bump, resulting in $n-1$ bumps. In turn, the rescheduling of the $n-1$ bumped variables could result in bumps of $n-2$ variables each, for a total of $(n-1)(n-2)$ bumps. Assuming k is the minimum number of participants for any variable, the following formula gives the maximum number of bumps possible when scheduling a variable with n participants and all agents use the *NumParticipants* bumping strategy:

$$\textit{Bumps}(n) = \sum_{i=0}^{n-1-k} \prod_{j=0}^{i} (n-1) - j \qquad (1)$$

For example, if $n = 4$ and every variable has at least 2 participants ($k = 2$), then the maximum number of bumps possible is $3 + 3 \times 2 = 9$. An attractive feature of this strategy is that it provides an upper bound on the amount of schedule disruption that may occur.

3.2.3 Scheduling Difficulty (SD)

This strategy assumes that each agent is given or has built a model from experience of other agents' ability or willingness to accept proposals. For example in the CMRadar domain discussed later, each agent is operating on behalf of a human so the model could take into account the stubbornness of the other agent or its promptness of reply. To be computationally convenient, we represent this model as a single number called a "scheduling difficulty" factor. In this paper, we will use average schedule density as the scheduling difficulty factor.

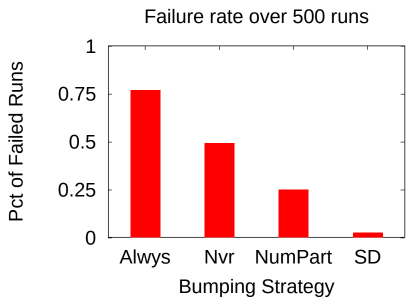

Figure 2: Comparison of bumping strategies in a four level organization hierarchy of 32 agents.

Let *Difficulty*(V_i) be a number denoting the *scheduling difficulty* of a variable V_i, i.e., if *Difficulty*$(V_i) >$ *Difficulty*(V_j), then finding a consistent value for V_i is expected to be more "difficult" than V_j. Concretely, we calculate scheduling difficulty by correlating it with the probability that a proposed value is unassigned in all participants current schedules. Let $Den_i = \mid Vars(A_i) \mid \div \mid \mathcal{D} \mid$ be the *schedule density* of an agent A_i. If A_1,A_2,...,A_k are the participants in variable V_i, and Den_1,Den_2,...,Den_k are the participants respective schedule densities, we calculate the scheduling difficulty of V_i as:

$$\textit{Difficulty}(V_i) = (1 - Den_1) \times (1 - Den_2) \times ... \times (1 - Den_k)$$

For example, if A_1 and A_2 are participants in variable V_i, with schedule densities of .9 and .4 respectively, then the probablity that a given value is unassigned by both participants is calculated as *Difficulty*$(V_i) = (1 - 0.9) \times (1 - 0.4) = 0.06$. This is an approximation because it assumes that A_1 and A_2 have independent schedules, which may not be strictly true.

Finally, the bumping strategy is defined as follows.

- **SD Strategy:** If a proposal (V_i, d_i) conflicts with current assignment (V_j, d_j), accept the proposal if and only if *Difficulty*$(V_i) >$ *Difficulty*(V_j).

The intuition is that variables with less constrained participants are easier to reschedule and so should be bumped in favor of variables with highly constrained participants.

4. EXPERIMENTAL RESULTS

An important component for development of DCR techniques is evaluation in realistic testbeds or on realistic benchmarks. Most existing work has focused on the use of abstract problems such as distributed graph coloring. We evaluate our techniques in the context of the *CMRadar Project* [13] whose goal is to develop personalized assistant agents that are able to make people more efficient by automating many routine tasks such as meeting scheduling.

In this section, we first describe a simulator for generating multiagent meeting scheduling problems. Then, we use the distributed CMRadar agents to execute the Iterative Agreement Protocol with different bumping strategies and present our experimental results in this domain.

4.1 Experimental Testbed

We evaluate each strategy over a number of *runs*. Each run consists of two phases, a) a centralized problem generation phase and b) a distributed problem solving phase. We describe each phase in turn.

Problem Generation The problem generation phase has three steps. In step one, we generate a set of CMRadar agents with empty calendars but each with a desired schedule density as specified by an input parameter. Each agent's calendar has 50 timeslots to simulate a 5 day, 10-hr/day work week. In step two, we repeatedly generate a meeting between a random subset of the agents, choose a random mutually free timeslot, and insert the meeting into the calendars. We continue until all calendars are filled to their desired density. The number of attendees for each meeting is chosen according to a distribution in which meetings of more people are less likely than meetings with fewer people, and every meeting has at least two attendees. In step three, we generate one additional new meeting M_{m+1} that must be scheduled in Phase 2. The attendees of meeting M_{m+1} are chosen to be a random subset of the agents, with the size of the meeting as an input parameter. One of them is randomly chosen to be the initiator.

Problem Solving The problem solving phase is completely distributed. The CMRadar agents live in a simulated distributed environment and are able to pass simulated email messages between them. Their goal is to find a timeslot for the new meeting $\{M_{m+1}\}$ while successfully rescheduling any bumped meetings. That is, the goal is to find a valid and complete assignment of timeslots to meetings. We measure the number of *failed runs* defined as a run in which this goal is not achieved after a given amount of time. Failures occur either because the initiator gives up on scheduling the meeting or a max time elapses.

4.2 Experiments in a Hierarchical Agent Organization

Human organizations typically have hierarchies in which higher ranked people have denser calendars than lower ranked ones. We experiment with an organization with four levels with 8 agents in each level, for a total of 32 agents. The four levels have initial schedule densities of 90,70,50,30 percent respectively. All agents use the same strategy. The size of the new meeting was fixed to four agents. The empirical results over 500 runs for each strategy are shown in Figure 2.

The Always strategy fails to schedule all the meetings in most cases. Closer inspection reveals that every failure is due to the expiration of the max time limit. We see the Always strategy is undesirable (at least when all agents employ it simultaneously) and a more discretionary strategy is needed. The Never strategy does slightly better, but still fails in roughly half the cases. The slightly more informed strategy *NumParticipants* does still better with a failure rate of 0.28. Finally, the failure rate is reduced to 0.02 using the SD strategy. We conclude that the SD strategy significantly reduces the number of scheduling failures in our experiments. These results demonstrate how agents are able to use additional knowledge of other agents (their schedule densities) to make more effective local scheduling decisions.

Next, we examine the disruption caused by each bumping strategy. Figure 3 shows histograms depicting the number of *bumps* for each strategy. Each histogram has on its y-axis the number of runs out of 500 which had the given number of bumps shown on the x-axis. Figure 3 (a) shows that the Always strategy results in a significant number of bumps; over 350 runs had greater than 16 bumps. Figure 3 (b) shows that the Never strategy has zero bumps for all 500 runs, as should be expected for this strategy. Figure 3 (c) and (d) show the schedule disruption for the *NumParticipants* strategy and SD strategy, respectively. Both strategies perform comparably, although the SD strategy has a slightly higher average. We notice that the $NumParticipants$ strategy is often significantly less than 9 as computed by Equation 1 in this case where the new meeting has 4 participants.

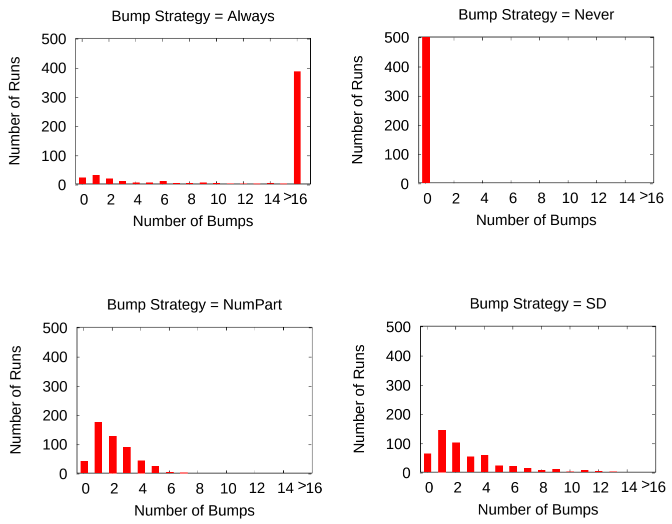

Figure 3: Measuring schedule disruption for four bumping strategies.

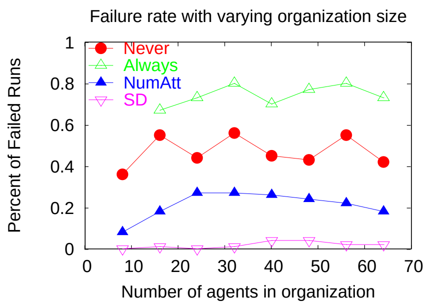

Figure 4: Comparison of bumping strategies with varying organization size.

Finally, Table 1 shows for each strategy the average number of rounds, messages and number of timeouts out of the 500 runs. As shown in the third column, the Always strategy results in uncontrolled bumping until a max time limit is reached. Note also that the *NumParticipants* and SD strategy require similar amounts of rounds and messages. This is significant because it shows that the reduction in failure rate shown in Figure 2 is obtained without a decrease in efficiency.

4.3 Varying Organization Size and Meeting Size

Figure 4 contrasts the strategies as we increase the size of the

Table 1: Number of rounds, msgs, and timeouts for four bumping strategies.

Strategy	Avg Rounds	Avg Msgs	NumTimeouts/NumRuns
Always	300	2843	384/500
Never	13	80	0/500
NumParticipants	9.57	38	0/500
SD	10.41	49	0/500

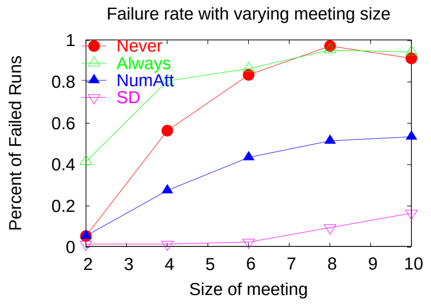

Figure 5: Comparison of bumping strategies with varying size of new meeting.

organization. Each datapoint represents the average of 100 runs. We see scale up to organization sizes of up to 64 agents. The qualitative results from the previous section in which the SD strategy outperforms, continue to be seen.

Figure 5 contrasts the strategies as we increase the number of attendees in the new meeting to be scheduled. Each datapoint represents the average over 100 runs. We see that meetings of up to 10 agents are able to be scheduled with a high likelihood of success (failure rate = 20%).

5. RELATED WORK

There has been significant research on meeting scheduling, but only a subset of this research has considered the problem as inherently decentralized. Of this subset, very few works have focused explicitly on its incremental aspect and the consequent bumping problem. Indeed, effective strategies for deciding when to reschedule meetings is lacking in previous distributed meeting scheduling research.

We classify related research into three categories, multiagent meeting scheduling, distributed scheduling in other domains, and distributed constraint reasoning.

5.1 Multiagent Meeting Scheduling

Sen and Durfee have done extensive work in multiagent meeting scheduling [16, 17]. They formalize the multiagent meeting scheduling problem and identify a family of negotiation protocols aimed at searching for feasible solutions in a distributed manner[17]. They also describe a contract-net approach for multiagent meeting scheduling [16] in which rescheduling and cancellation of existing meetings is briefly discussed. However, rescheduling of existing meetings or modeling of other agents to improve performance has not been a major focus.

Freuder, Minca and Wallace [8] have previously investigated meeting scheduling within the Distributed Constraint Reasoning framework. Their work is notable for empirically demonstrating a privacy/efficiency tradeoff in multiagent meeting scheduling. Ephrati and collegeaues have taken an economic approach to scheduling in which agents express preferences for meeting times using a monetary "points" system [6]. Their approach assumes existence of a centralized scheduler with a global view of all calendars, although user preferences are distributed and may be kept private.

5.2 Distributed Scheduling

Previous research in distributed scheduling has focused on a variety of domains including job-shop scheduling [19], airport scheduling [3] [14] and medical scheduling [10] [5]. One of the more influential ideas from previous research in distributed scheduling has been the communication of high-level information called texture measures [19]. In this approach, agents coordinate their scheduling decisions by communicating high-level information about their local scheduling problem, such as their demand for a resource, so that agents can ensure resources are allocated to the most constrained agents. The scheduling difficulty models introduced in Section 3.2 can be viewed as an instantiation of this heuristic approach.

5.3 Distributed Constraint Reasoning

There exists some work in DCR dealing explicitly with privacy, most notably the work of Yokoo et al. [21] and Silaghi et al. [18]. These approaches have focused on the use of strong cryptographic techniques such as using homomorphic encryption functions to encode communicated information. Yokoo et al. present a Secure DisCSP algorithm that provides privacy guarantees and theoretical guarantees on algorithm completeness.

6. CONCLUSION

We have modeled an important class of scheduling problems as a form of DCR in which multiple agents must assign a set of values to a set of variables according to local intra-agent constraints and external inter-agent equality constraints. We presented one of the first informed heuristic approaches to DCR in which agents use given scheduling difficulty models of other agents in order to decide when to modify existing assignments. We show that this approach reduces the scheduling failure rate and controls the amount of schedule disruption. In future work, we are interested in developing theoretically complete techniques for piMAP and also focusing in how useful heuristic information about other agents can be learned through experience.

Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DARPA or the Department of Interior-National Business Center (DOI-NBC).

7. REFERENCES

[1] J.C. Beck, A.J. Davenport, E.M. Sitarski, and M.S. Fox. Texture-based heuristics for scheduling revisited. In *Proceedings of AAAI-97*.

[2] C. Bessire, A. Maestre, and P. Meseguer. Distributed dynamic backtracking. In *IJCAI Workshop on Distributed Constraint Reasoning*.

[3] M.H. Chia, D.E. Neiman, and V.R. Lesser. Coordinating Asynchronous Agent Activities in a Distributed Scheduling System. In *Proceedings of International Conference on Multi-Agent Systems*, January 1998.

[4] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[5] K. Decker and J. Li. Coordinated hospital patient scheduling. In *Proceedings of International Conference on Multi-Agent Systems*, 1998.

[6] Eithan Ephrati, Gilad Zlotkin, and Jeffrey S. Rosenschein. A non–manipulable meeting scheduling system. In *Proceedings of the 13th International Workshop on Distributed Artificial Intelligence*, Seatle, WA, 1994.

[7] B. Faltings and S. Macho-Gonzalez. Open constraint satisfaction. In *Principles and Practice of Constraint Programming - CP 2002*, pages 356–370, 2002.

[8] E. C. Freuder, M. Minca, and R. J. Wallace. Privacy/efficiency tradeoffs in distributed meeting scheduling by constraint-based agents. In *IJCAI-2001 Workshop on Distributed Constraint Reasoning*, 2001.

[9] L. Garrido and K. Sycara. Multi-agent meeting scheduling: Preliminary experimental results. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*. The MIT Press: Cambridge, MA, USA.

[10] M. Hannebauer and S. Mller. Distributed constraint optimization for medical appointment scheduling. In *Proceedings of the Fifth International Conference on Autonomous Agents*, 2001.

[11] R. Mailler and V. Lesser. A mediation based protocol for distributed constraint satisfaction. In *The Fourth International Workshop on Distributed Constraint Reasoning*, 2003.

[12] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 2004.

[13] P. J. Modi, M. Veloso, S. Smith, and J. Oh. Cmradar: A personal assistant agent for calendar management. In *Agent Oriented Information Systems, (AOIS)*, 2004.

[14] D. Neiman, D. Hildum, V. Lesser, and T. Sandholm. Exploiting Meta-Level Information in a Distributed Scheduling System. In *AAAI*, 1994.

[15] J. Oh and S.F. Smith. Learning user preferences for distributed calendar scheduling. In *Proc. 5th International Conference on Practice and Theory of Automated Timetabling (PATAT)*, Pittsburgh, PA, 2004.

[16] Sandip Sen and Edmund Durfee. A Contracting Model for Flexible Distributed Scheduling. *Annals of Operations Research*, 65:195–222, 1996.

[17] Sandip Sen and Edmund H. Durfee. A formal study of distributed meeting scheduling. In *Group Decision and Negotiation*, volume 7, pages 265–289, 1998.

[18] M.C. Silaghi and D. Mitra. Distributed constraint satisfaction and optimization with privacy enforcement. In *3rd IC on Intelligence Agent Technology*, 2004.

[19] K. Sycara, S. Roth, N. Sadeh, and M. S. Fox. Distributed constrained heuristic search. *IEEE Transactions on Systems, Man, and Cybernetics*, 21:1446–1461, 1991.

[20] M. Yokoo. *Distributed Constraint Satisfaction:Foundation of Cooperation in Multi-agent Systems*. Springer, 2001.

[21] M. Yokoo, K. Suzuki, and K. Hirayama. Secure distributed csp: Reaching agreement without revealing private information. In *Constraint Programming*, 2002.