1998 International Computer Symposium
Workshop on Artificial Intelligence
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

NEGOTIATION STRATEGIES IN DISTRIBUTED MEETING SCHEDULING USING MOBILE AGENTS

Huang Ho Shu An and Von-Wun Soo

Department of Computer Science

National Tsing-Hua University, Hsin Chu, Taiwan, R. O. C.

E-mail: soo@cs.nthu.edu.tw

Abstract

Distributed meeting scheduling can cause a problem using simple contract net protocol. We design a distributed meeting scheduling systems using mobile agent and propose negotiating strategies of the agents in order to relieve the problem. But introducing negotiating efficiency of scheduling. In order to find ways of decreasing the number of negotiations, we conducted experiments on such negotiation strategies as random, minimum-member, shortest meeting length, and the lowest preference respectively and found that the shortest meeting length strategy would usually achieve meeting scheduling with less number of negotiations.

1 Introduction

1.1 A distributed meeting scheduler

It is tedious and time-consuming for people to do the meeting scheduling task. Although many commercial products have been developed to solve the task, they do not solve the meeting scheduling problems in distributed environment. Because the popularity of Internet, we have more opportunities to schedule meeting with other people. For example, in teleconferencing, it is important to use a distributed meeting scheduler to schedule a meeting before it is actually held. Because in distributed environment there's no server to schedule meetings. All attendees need to negotiate to come up with a solution. To negotiate, attendees need a protocol to exchange information. Some distributed meeting scheduling systems [2] use a following simple contract-net protocol [1]

1. A host agent tries to find time intervals available in its schedule. The host agent announces a contract for the meeting to the invitees by proposing one or more of the time intervals available.
2. Each invitee receives the contract proposal, and tries to find local solutions to satisfy those contracts and send them back as bids to the host. Bids consist of time intervals for which the bidder (the invitee) can attend the announced meeting. The time intervals sent as bids can simply be the subset of those announced by the host that the invitee has free on its calendar, or they can be counter-proposals for another time interval to meet.
3. The host collects and evaluates these bids. If the bids suggest a common time interval which is free for the host as well, the meeting can be scheduled and the host sends confirmations to the bidders. If the meeting cannot yet be scheduled, the host sends new proposals depending on the bids received and its own calendar to the bidders. It also sends rejections for bids received.

4. When the bidders receive new proposals, they reply as above. On receiving confirmation, they check to see if those time intervals are still free. If so, they mark their calendar, recording the scheduling of the meeting. Otherwise they send rejections back.

The above protocols are repeated until a meeting is achieved or it is recognized that the meeting scheduling fails.

1.2 Problem description

In fig.1, a flowchart of a simple contract-net protocol for meeting scheduling is shown.

A Scenario: For attendee A, B, C, D, E, F. Assume A, E and F have free time 2 and 3. B and C have free time 1, 2, 3, and D has free time 2 respectively as shown below. Let○ denote a free time slot and ● denote a blocked time slot.

	A	B	C	D	E	F
Time slot1	○	○				
Time slot2	○	○	○	○	○	○
Time slot3	○	○	○		○	○

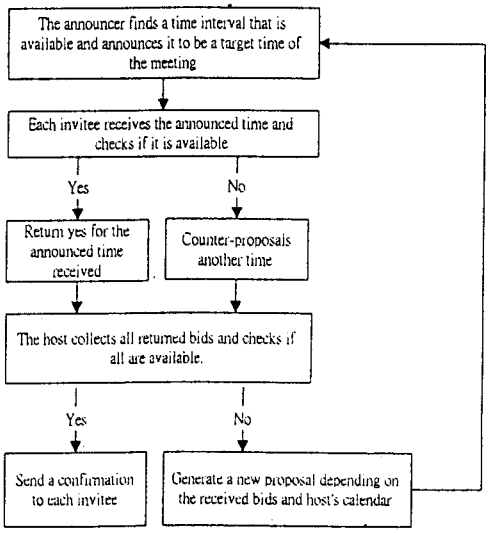

Fig 1. A flowchart of a distributed meeting scheduling protocol

Now, assume 4 meetings are to be scheduled: M1={B, C} at time slot2; M2={A, C, D} at time slot2; M3={A, C, E} at time slot3; M4={B, E, F} at time slot3.

The time table after scheduling M1 is shown below:

	A	B	C	D	E	F
Time slot1	○	○				
Time slot2	○	●	●	○	○	○
Time slot3	○	○	○		○	○

If M1 is scheduled at time slot2, we find that there's no free time for M2 to be scheduled. So the system will inform all attendees (A, C, D) in M2 that M2 can't be scheduled. Continue working with M3 and M4, we find that M3 can be scheduled at time slot3 but not M4.

	A	B	C	D	E	F
Time slot1	○	○				
Time slot2	○	●	●	○	○	○
Time slot3	●	○	●		●	○

Without rescheduling, the above 4 meetings can not be scheduled completely. With rescheduling, all of the above will be scheduled although they may not fit the highest preference criterion.

The following is a possible scheduling condition for the 4 meetings above to be scheduled completely. The time table conditions after scheduling M2 is shown below:

	A	B	C	D	E	F
Time slot1	●	●				
Time slot2	●	○	●	●	○	○
Time slot3	○	○	○		○	○

The time table conditions after scheduling M3 is shown below:

	A	B	C	D	E	F
Time slot1	●	●				

Time slot2	●	○	●	●	○	○
Time slot3	●	○	●		●	○

The time table conditions after scheduling M4 is shown below:

	A	B	C	D	E	F
Time slot1	●	●				
Time slot2	●	●	●	●	●	●
Time slot3	●	○	●		●	○

M1 at time slot1, M2 at time slot2, M3 at time slot3, M4 at time slot2

So, we need a new protocol to achieve the situation. A modified protocol will be discussed later.

1.3 Mobile Agents

The architecture of a mobile agent is the same as that of a traditional agent. But it is an important feature on Internet. Because Internet is a fully distributed environment, we must use powerful agents to get more resources. The following is the advantages of a mobile agent in comparison to traditional agent.

Object-passing : When a mobile agent moves, the whole object is passed; it means that the code, execution state, traveling itinerary are passed too.

Autonomous : Some traditional agents include this feature too, but a mobile agent contains sufficient information to decide what to do and where to go.

Local interaction : A mobile agent interacts with other agents or stationary objects locally.

Disconnected operation : A mobile agent can do his job whether the network is connected or not. If the network connection is broken, a mobile agent needs to move to another place. And it will wait until the connection is reconnected.

1.4 Voyager - A mobile agent platform

Voyager [7] is a Java-based language to develop mobile agent. It is a powerful platform for creating distributed Java application. Agent platforms such as Odyssey, Aglets [6], and Concordia provide the ability to create, program and launch an agent into networks. Aglets uses sockets and Odyssey or Concordia use RMI to move agents between machines, but neither allow you to send a regular Java message to a stationary or moving agent. This means that it is very hard to communicate with an agent after it has been launched, and very hard for agents to communicate directly with other agents.

We adopt Voyager as a platform of our mobile agents due to the several reasons:

1. In Voyager, it leaves behind a trail of forwarders that will forward messages to the object's new location. So it is easily to send message to remote objects.
2. The Java system supports the concept of security managers, which are installable "watchdogs" that prevent unauthorized code from executing a preset variety of operations.
3. Many distributed systems require facilities for communicating with groups of objects.
4. The locations to visit can be stored as a Vector of addresses and an agent can move easily from location to location, performing tasks as it goes.

2. The Distributed Meeting Scheduling System

Using a centralized meeting scheduling system, it is less private and secure than distributed meeting scheduling system. Besides, the centralized scheduling system needs to collect time tables of all attendees to solve the

scheduling problem, it is not practical in reality. To design a distributed scheduling system, we need to deign a multi-agent environment [5] using the mobile agents platform discussed above, but also new negotiation and scheduling protocols in order to avoid the problem in simple contract-net protocol addressed earlier.

2.1 The main scheduling protocol

If (the requested time intervals are free)
{ If (no other time interval)
{ Return (OK and blocked) }
Return (OK and the requested time interval)
}
Elseif (conflict()) //conflict() is for judging all time intervals are conflicted
Negotiate()
Else Counterpropose() //Depends on the table of preference

2.2 The negotiation protocol

The data structure and procedure used in a meeting scheduler is:

- Scheduling element, namely a meeting, is a tuple <Attendee, time, status> in the *history record* to be discussed later.
- Scheduling list consists of a list of scheduling elements: <SL> = < e_1, e_2, ... e_n>
- Negotiable list consists of a list of negotiable meetings<NL> = < n_1, n_2, ... n_m> where <n_1, n_2, ..., n_m >⊆ <e_1, e_2, ..., e_n>

Find suitable negotiable element sn∈ n_1, n_2, ..., n_m(*using the strategies to be discussed later*)
While (N(<NL>)>0)
{ If (negotiable(sn))
Schedule()
Else
{ <NL> = <NL> - sn
Find the next suitable negotiable element in <NL> }
}

2.3 Parameters for scheduling

There are many constraints in a meeting scheduling system. In our system, we concern only the time constraints and ignore for now the meeting room, meeting tools, and other constraints. Our goal is to schedule the maximum number of meetings within a time period. Meeting parameters can be meeting name, duration, deadline of scheduling, start time, attendee and meeting characteristics such as whether a meeting is blocked or negotiable. Others such as Preference of time is also an important parameter. Many researches are concerned with how to schedule a meeting with the highest preference (to all attendees) using case-based reasoning [4].

But in our system, the goal is to schedule more number of meetings using as less number of negotiations as possible. The satisfaction of user preference is regarded as the second level of meeting scheduling goal.

2.4 System architecture and description

Each user has two agents: a scheduling agent and a negotiating agent.

Scheduling agent: When a user wants to announce a meeting, he needs to fulfill the parameters described earlier. Then he uses the scheduling agent to schedule with other negotiating agents.

Negotiating agent: When a scheduling agent announces a time interval but other attendee's negotiating agent does not accept it, the negotiating agent needs to propose another time

interval. There are two ways for a scheduling agent to schedule a meeting:

1. Using original announcer's scheduling agent continually to schedule the meeting.
2. Using the scheduling agent of an attendee who does not accept the proposed time interval to continue the scheduling task.

We choose the former in our system. The reason is because the same scheduling agent can record the scheduling history and thus can avoid the redundancies of the scheduling history.

2.5 Negotiation history record

When a scheduling agent and negotiating agent negotiate to schedule a meeting, no matter what decision the negotiating agent makes, the scheduling agent records it. The format is below:

User name: record the name of the user for whom the answering agent is worked and number of encounters of scheduling agent and negotiating agent.

Time: Suggested time Ex. Apr 11 12:00 1998

Commitment: This will discussed later.

Meeting priority: A meeting announcer can judge the importance of a meeting. Then he can assign the status of a meeting as negotiable or blocked. Negotiable meeting means that the meeting has a lower priority and the blocked meeting has a higher priority. Each attendee who accepts the proposed time must modify his time status.

3. A scenario of solving the previous problem

We test with the problem that happened in a distributed meeting scheduling system with a central host.

The notations ○, #, X represent three time statuses of free, negotiable, blocked respectively. Suppose all meetings are negotiable (if all are blocked, the results will be the same as before.

1. When scheduling M1= {B, C}

History record

1 B2○(nego)C2○(nego)

M1 is scheduled at time slot2

	A	B	C	D	E	F
Time slot1	○	○				
Time slot2	○	●	●	○	○	○
Time slot3	○	○	○		○	○

2. When scheduling M2 = { A, C, D }

History record

1 A2○(nego) C2#

2 C1○(nego)D1X

It needs negotiation at this situation because M1 is negotiated to time slot1 or time slot3 depending on the preference of (B,C). Now we suppose negotiate to time slot3.

M1 is scheduled at time slot3 and M2 is scheduled at time slot2.

	A	B	C	D	E	F
Time slot1	○	○				
Time slot2	●	○	●	●	○	○
Time slot3	○	●	●		○	○

3. When scheduling M3 = {A, C, E}

1A3○(nego) C3#

2C1○(block) B1○(block) E1X

It needs negotiation on this situation because M1 is scheduled to time slot1 and M3 is scheduled to time slot3

	A	B	C	D	E	F
Time slot1	●	●				
Time slot2	●	○	●	●	○	○
Time slot3	●	○	●		●	○

4. When scheduling M4 = {B, E, F}

1B3○(nego) E3X

2E2○(block) B2○(block) F2○(block)

M4 is scheduled at time slot2.

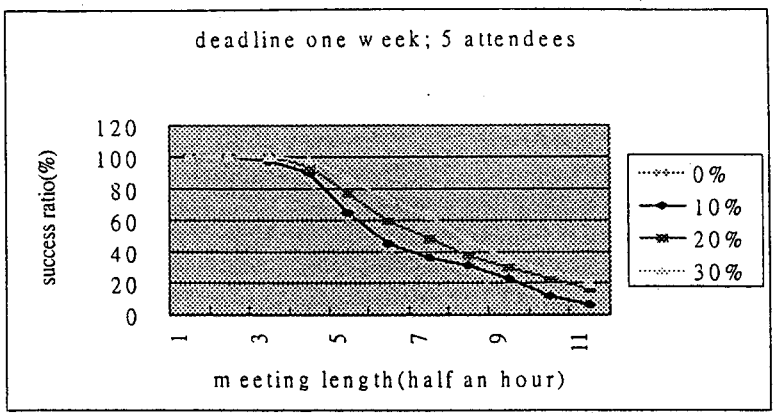

Fig. 2

	A	B	C	D	E	F
Time slot1	●	●				
Time slot2	●	●	●	●	●	●
Time slot3	●	○	●		●	○

All 4 meetings were scheduled successfully.

4. Commitment and Negotiation Strategies

4.1 Commitment strategies

When a negotiating agent receives a proposed time. He needs to reply his commitment to the scheduling agent. There are three conditions of the proposed time, but there are 7 possible ways that the negotiating agent can commit. The three conditions of the proposed time can be: blocked, free and negotiable.

4.2 Negotiation strategies

When entering a negotiation mode, it could be at least one meeting that needs to be negotiated or a new incoming meeting not yet be scheduled. From the history record of meeting we described, we need to select one from the negotiation list to negotiate until all negotiating elements fail. Three strategies plus a random strategy are used in our experiments:

- **Choose the meeting with minimum number of attendee.**
- **Choose the lowest preference of meetings.**
- This strategy does not concern with how to decrease the number of negotiations. It just attempts to increase the average meeting preference.
- **Choose the meeting with the shortest meeting length**

5. Experimental results and evaluation

We design experiments to test the distributed meeting scheduling system. In all experiments, the time table is supposed to be from 9:00 a.m. to 5:00 p.m. There are seven days in a week. In the preference table, user can have preferences on time (9:00 a.m. to 5:00 p.m.) or days (Sunday to Saturday). The minimum time interval is 30 minutes. It means that meeting length is not less than 30 minutes.

Experiment 1:

In this experiment, we compare the relations between the success ratio and the meeting length. The time table distribution of all attendee is 50% free and a varying percentage of the negotiable time. In Fig 2, there are 4 curves whose negotiable times are 0%, 10%, 20%, 30%, respectively. We find that the success ratio declines when meeting length is increased. However, we found in the figure when we increase the percentage of negotiable time, the

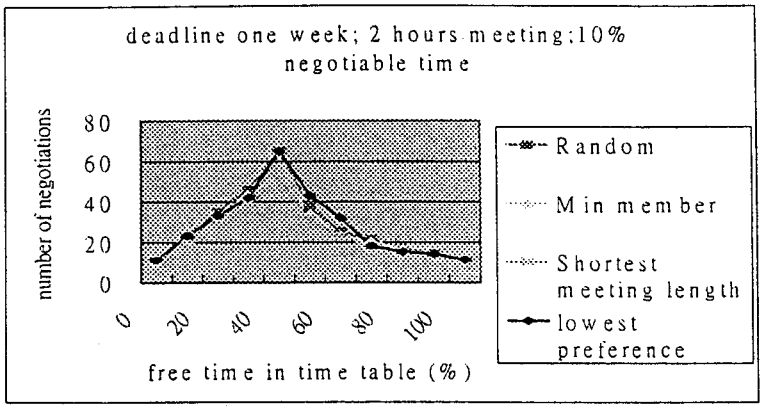

Fig. 3

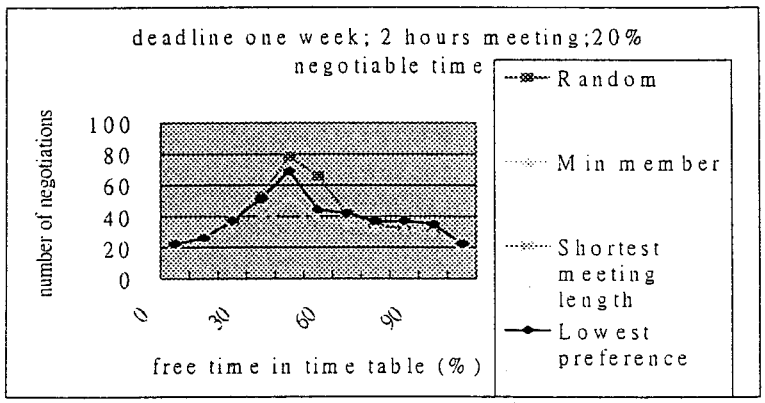

Fig. 4

success ratio increases. It means that we can schedule more meetings in our system

Experiment 2:

In this experiment, each meeting contains one to ten attendees randomly. The figures 3 and 4 are the results of testing the number of negotiations against percentage of free time in time table. We found that the peak number of negotiations appeared at about 45%-55% of free time. The reason is that if each user's percentage of free time in the time table is very low or very high, the scheduling agent will finish scheduling a meeting either successfully or fails eventually. But when the percentage free time is about 45% to 55%, the scheduling agent needs more number of negotiations to achieve the task. The four curves in figure 4 are random, minimum member, shortest meeting length, lowest preference respectively. We find that using the shortest meeting length strategy in finding negotiable meeting is better than other in the sense that it requires least number of negotiations.

Experiment 3:

In fig 5, when each user has 30% negotiable time and 0% negotiable, the average preferences don't vary a lot when the meeting length increases, the rest of the 0% curve, because the scheduling did not succeed, no average preference is available.

5. Discussion

From the experiment 1 and 2, we find a trade-off between the number of negotiations and the success ratio upon percentage of free degree of negotiable time. According the result of experiment 2, the shortest meeting length

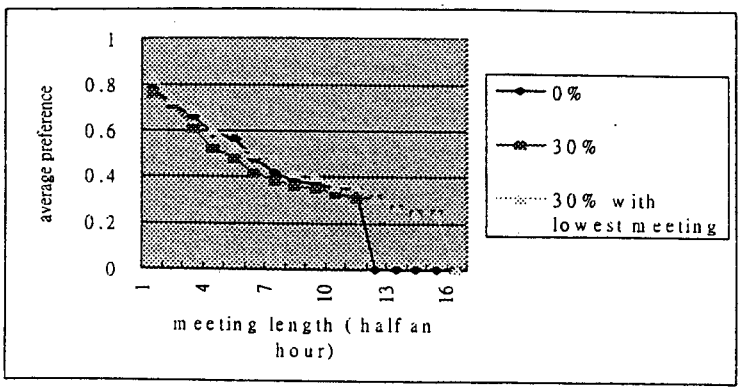

Fig. 5

strategy is the most efficient way to schedule a meeting when it is needed to choose a negotiable meeting. According the result of experiment 3, the strategy of finding the lowest preference meeting to negotiate is useful to get higher preference. Many parameters can be modified in meeting scheduling. If we change some parameters like the number of attendees or deadline of a meeting, the results might change too. There is no standard benchmark to verify distributed meeting scheduling systems, so each developer simulates experiments under his own assumption. For this reason, it is hard to compare the performance of different systems.

7. Conclusion and Future work

We have developed an agent-based meeting scheduling system that can schedule meetings via negotiations and a new protocol that avoids the problems in other distributed meeting scheduling systems. We adopted the idea of mobile agent to overcome the message passing and speed of communication problems. From the experience of implementation, we found that Voyager is a suitable platform to develop mobile agents. However, we seldom address the consideration of user's preference. Satisfying user's preference is an important issue [4]. We could include some strategies to get higher meeting preferences or adopt learning mechanisms to satisfy user's preference [3]. We could also consider the priority of attendee or allow a scheduling agent to learn the priority of scheduled meetings to each attendee. The concurrency of distributed database is also an important issue. We still need work on this part in the future.

Reference

[1] Smith, R.. *The contract net protocol: High level communication and control in distributed problems solver.* IEEE Transactions on Computers, 29(12):1104-1113, 1980.

[2] Sandip, S. and Durfee, E. H. *A formal study of distributed meeting scheduling: preliminary results.* In ACM Conference on Organizational Computing Systems, 1991.

[3] Mitchell, T.; Caruana, R.; Freitag, D.; McDermott, J.; and Zabowski, D.. *Experience with a learning personal assistant.* Communication of the ACM37(7):80-91, 1994.

[4] K. Miyashita and K. Sycara. *Using Case-, based reasoning to acquire user scheduling preferences that change over time.* In CAIA. IEEE, 240-246, 1995.

[5]Sycara, K. and Garrido L.. *Multi-agent meeting scheduling preliminary experimental results.* ICMAS. 95-102, 1996

[6]Programming mobile agent in Java, IBM Tokyo Research Lab, http://www.trl.co.jp/aglets.

[7]Objectspace, http://www.objectspace.com