THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學

Department of Computing

Design and Implementation of ASSIST: Automatic Secretary System using Internet and Smart-agent Technologies

Irene Sze-Kwan Ho

A Thesis Submitted in Partial Fulfillment
of the Requirements for the
Degree of Master of Philosophy

February 2004

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written nor material which has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

Irene Sze-Kwan Ho

ABSTRACT

Abstract of thesis entitled 'Design and Implementation of ASSIST: Automatic Secretary System using Internet and Smart-agent Technologies' submitted by Ho Sze Kwan for the degree of Master of Philosophy at the Hong Kong Polytechnic University in February 2004.

With the advent of various Internet technologies, it is now feasible and cost-effective to provide secretarial services by electronic means in general and by software agents in particular. This thesis presents an automatic secretary system called ASSIST (Automatic Secretary System using Internet and Smart-agent Technologies). Generally, ASSIST provides basic secretarial services such as meeting scheduling, email management and document management for professionals and executives. Like other electronic secretarial systems, our focus is on the scheduling of meetings.

Inspired by other similar systems, the meeting scheduling mechanism basically works as follows. Upon receiving a request, some possible time slots are found based on information about the meeting. The system then sends these time slots to the invitees enquire as to their availability. The meeting can be arranged if a mutually agreed time slot is found. If not, the above process is repeated. In most of the existing systems, an arbitrary number of time slots are used for scheduling purposes. In this thesis, we present two analytical models for calculating the number of suggested time slots so that the average cost of scheduling a meeting

can be minimized. Based on the first model, the best number of time slots per scheduling cycle can be determined. As an extension to the first model, the second model is Markov-decision-based, which takes into account the scheduling deadline and other meeting parameters. By using this model, the number of suggested time slots can be varied dynamically. For both models, we have conducted extensive analysis to evaluate their performance.

Finally, we have investigated some matching algorithms for finding a common time slot among some proposed time slots. These algorithms are the random matching algorithm, ascending availability matching algorithm, descending availability matching algorithm, simplified ascending availability matching algorithm and simplified descending availability matching algorithm. We have compared the aforementioned algorithms by using computer simulations. It was found that the ascending availability matching algorithm provides the best performance in general in terms of the scheduling cost and the successful rate for finding a time slot.

ACKNOWLEDGEMENTS

I would like to take this opportunity to give my sincere thanks to my supervisor, Dr. Henry C.B. Chan. He has given me valuable suggestions, advices and comments on my study, research, as well as my career. He spent a lot of time on my research project, although he was very busy on other projects and teaching. Also, I would like to thank my co-supervisor Dr. Stephen C.F. Chan for his valuable comments and suggestions on my research project.

In addition, I would like to thank The Hong Kong Polytechnic University for providing me with a good environment for my study and granting me the Tuition scholarships for Research Postgraduate Studies. This project has been supported by the Innovation and Technology Fund/Teaching Company Scheme and Macroview Telecom Ltd. I would like to thank Macroview Telecom Ltd. for their valuable advices and support throughout the project.

I would like to thank Mr. Gary Li for conducting some preliminary work related to this project, particularly the virtual desktop and a basic meeting scheduling model. Part of my work in this thesis is based on the extension of his work. Also, I would like to thank the final year project students: Ms. Sharon Yu, Mr. Alex Cheung, Mr. Eddie Fok, Mr. Warren Fok and Mr. Wilson So who have participated in developing the prototype system.

Last but not the least, I would like to give special thanks to my family and friends who have supported me throughout the years. I hope all of them have a healthy life and bright prospects.

Table of Contents

ABSTRACT II

ACKNOWLEDGEMENTS IV

CHAPTER 1 INTRODUCTION 1

1.1 MOTIVATION 1
1.2 SCOPE & OBJECTIVES 3
1.3 ORGANIZATION 5

CHAPTER 2 BACKGROUND STUDY 7

2.1 RELATED WORK ON VIRTUAL SECRETARY PROJECTS 7
2.2 RELATED WORK ON MEETING SCHEDULING 17
2.2.1 Summary of the meeting scheduling 52
2.3 MOBILE AGENT INTRODUCTION 53

CHAPTER 3 ASSIST 55

3.1 SYSTEM ARCHITECTURE OF ASSIST 56
3.2 ASSIST'S FUNCTIONS 57
3.3 COMMUNICATION BETWEEN ASSISTS 65
3.4 SUMMARY 75

CHAPTER 4 MEETING SCHEDULING 76

4.1 SCHEDULING WORKFLOW 76
4.2 MATHEMATICAL MODEL 81
4.2.1 Average Cost of Scheduling 82
4.2.2 Probability Q (probability that the meeting will be scheduled successfully) 87
4.3 RESULTS AND FINDINGS 89
4.4 SUMMARY 98

CHAPTER 5 MARKOV DECISION MODEL FOR SCHEDULING 99

5.1 MARKOV DECISION MODEL 101
5.1.1 Probability Q (probability that the available time slot(s) can be found) 106
5.1.2 Probability R (the probability that the sufficient number of invitees reply to the system) 106
5.1.3 Backward induction method to find the optimal number of proposed time slots 107
5.2 ANALYTICAL RESULTS AND DISCUSSIONS 108
5.3 SUMMARY 119

CHAPTER 6 120

MATCHING ALGORITHMS 120

6.1 MATCHING MODELS 120
6.2 ANALYTICAL RESULTS AND DISCUSSIONS 133
6.3 SUMMARY 152

CHAPTER 7 CONCLUSION 153

REFERENCES 156

SYMBOLS 162

Table of Figures

Figure 1. The system architecture of the software secretary proposed by Park and Kwak 9

Figure 2. Data flow for handling a request 10

Figure 3. System modules of the ViSe 11

Figure 4. Data flow when ViSe handles a request 14

Figure 5. Detailed protocol of request handling 14

Figure 6. The simplified system architecture of the software secretary 15

Figure 7. System components of the scheduling system 18

Figure 8. System architecture of the agent-based meeting scheduling system 23

Figure 9. Scheduling protocol of the system 25

Figure 10. System architecture of the meeting scheduling system 30

Figure 11. The relationship of facilitator, MA and PAs in a meeting scheduling system 30

Figure 12. Summary of the meeting protocol 32

Figure 13. Detailed protocol in the contract net protocol phase 33

Figure 14. Detailed protocol in cooperation phase 34

Figure 15. Detailed protocol in rescheduling phase 35

Figure 16. Detailed protocol in non-committed phase 36

Figure 17. Detailed protocol in committed phase 37

Figure 18. System architecture of the meeting scheduling system 41

Figure 19. The workflow of the scheduling system 43

Figure 20. Protocol in the generative state 44

Figure 21. Protocol in the announcing state 44

Figure 22. Protocol in the receive_bid state 45

Figure 23. Protocol in the evaluating state 46

Figure 24. Protocol in the negotiate state 47

Figure 25. Protocol in the award state 47

Figure 26. Example on calculating the number of time slots 49

Figure 27. Overview of ASSIST 55

Figure 28. Logic flow of the meeting scheduling function 64

Figure 29. Overview of the system architecture 65

Figure 30. Meeting scheduling workflow of a standalone ASSIST 66

Figure 31. Communication between two ASSISTs 67

Figure 32. DTD of RequeestMeeting.xml 68

Figure 33. Example of RequestMeeting.xml 69

Figure 34. DTD of Announcement.xml 70

Figure 35. Example of Announcement.xml 72

Figure 36. DTD for Preference.xml 74

Figure 37. Example of Preference.xml 75

Figure 38. Stage diagram for meeting scheduling 76

Figure 39. Schedules of A, B, C and D 77

Figure 40. Two time slots from 10:00 to 12:00 selected by the scheduling system 78

Figure 41. Selected time slots in the second iteration of the meeting scheduling 79

Figure 42. The two time slots selected in the third iteration of the scheduling 79

Figure 43. Eight time slots selected in the first iteration 80

Figure 44. Stage diagram to show that the meeting scheduling process 84

Figure 45. The server sends the proposed time slots to the users and the users give their replies 85

Figure 46. Optimal number of time slots/Average number of iterations with different availabilities 91

Figure 47. Optimal number of time slots with different meeting durations 91

Figure 48. Average number of iterations with different meeting durations 92

Figure 49. The optimal number of time slots with different overhead costs 93

Figure 50. Average number of iterations with different overhead costs 95

Figure 51. Optimal number of time slots with different attendance rates 96

Figure 52. Number of iterations with different attendance rates 97

Figure 53. Meeting scheduling procedures 100

Figure 54. State diagram for meeting scheduling model 103

Figure 55. State diagram for waiting state with n=4 104

Figure 56. Number of time slots with different availabilities 110

Figure 57. Number of time slots with different response rates 111

Figure 58. Number of time slots with different durations 112

Figure 59. Number of time slots with different overhead costs 113

Figure 60. Number of time slots with different filling costs/locking costs 114

Figure 61. Number of time slots with different terminating costs 114

Figure 62. Number of time slots with different attendance rates 115

Figure 63. Number of proposed time slots with different maximum number of time slots 117

Figure 64. Optimal time slot with different availabilities 117

Figure 65. Cost comparison between the dynamic and fixed time slot strategy 119

Figure 66. Number of iterations with different attendance rates 120

Figure 67. Random matching algorithm 122

Figure 68. Number of iterations with different attendance rates 123

Figure 69. Checking order based on the random matching algorithm 123

Figure 70. Ascending order on the availability of the invitees 125

Figure 71. Algorithm of the ascending-availability matching algorithm 126

Figure 72. Checking sequence using ascending-availability matching algorithm 126

Figure 73. Algorithm of the descending-availability matching algorithm 128

Figure 74. Descending order on the availability of the invitees 128

Figure 75. Checking order when using descending-availability matching algorithm 129

Figure 76. Checking order when using the simplified ascending-availability matching algorithm 130

Figure 77. Checking order when using the simplified descending-availability matching algorithm 131

Figure 78. Average scheduling cost with different maximum number of proposed time slots ... 134

Figure 79. Average number of cycles with different maximum number of proposed time slots. 135

Figure 80. Average success rate with different maximum number of proposed time slots 136

Figure 81. Average success rate with different matching algorithms 137

Figure 82. Average number of cycles with different matching algorithms 139

Figure 83. Average matching cost with different matching algorithms 141

Figure 84. Average success rate of different matching algorithms 141

Figure 85. Average number of cycles of different matching algorithms 143

Figure 86. Average number of cycles of different matching algorithms 144

Figure 87. Average success rate of different matching algorithms 145

Figure 88. Average number of cycles of different matching algorithms 146

Figure 89. Average matching cost of different matching algorithms 148

Figure 90. Average matching cost of different maximum number of proposed time slots 149

Figure 91. Average number of cycles of different maximum number of proposed time slots 150

Figure 92. Average success rate of different maximum number of proposed time slots 151

Table of Tables

Table 1. Detailed functions of the modules in ViSe *11*

Table 2. Responsibility of scheduler, communicator and calendar *41*

Table 3. Summary of the scheduling features *52*

Table 4 Detailed functions of ASSIST on the document management *57*

Table 5 Detailed functions of ASSIST on the email service *60*

Table 6. Elements and attributes in Announcement.xml *71*

Table 7. Elements and attributes used in Preference.xml *74*

Table 8. Notations used in the model *83*

Table 9. m, Q and cost in each iteration *85*

Table 10. Value of parameters used in the model *89*

Table 11. Notations used in the model *101*

Table 12. Transition probability *105*

Table 13. Scheduling cost *106*

Table 14. Value of the parameters used in the model *109*

Table 15. Comparison on the characteristic of different matching algorithms *131*

CHAPTER 1
INTRODUCTION

In this chapter, we give an overall of the research project and the thesis, including the motivation, scope, objectives and organization of the thesis.

1.1 Motivation

With the advent of Internet computing and various office automation systems [13], [27], [39], it is now cost-effective to provide automatic secretarial services.

Recall how a busy executive organizes his schedule if there is no personal secretary or assistant. He needs to schedule a meeting by himself. Firstly, he needs to define the information on the meeting, such as its purpose, its duration and the invitees. Secondly, he needs to check his personal organizer and searches for an available time slot for the meeting. Then he contacts the invitees one by one. If at least one invitee cannot attend the meeting at the proposed time slot, he needs to choose another time slot for the meeting based on his schedule. Subsequently, he needs to contact the invitees again. If the busy executive does not have an updated organizer, the confirmed meeting may be in conflict with other meetings. If the busy executive is too busy on his job and he has no time to check the organizer, he may forget to attend the meeting. If the meeting is very

important, the result and effect could be very serious. The above scenario shows how important a secretary is in assisting business executives.

The main tasks of a secretary or personal assistant are to organize the schedule of his/her boss, schedule meetings, receive phone calls, remind him/her to attend meetings, take the meeting minutes, file documents etc. It is not difficult to see that some of these jobs are repetitive and tedious, such as meeting scheduling and reminder service.

In recent years, a number of automatic secretary systems have been developed [4], [5], [7], [11], [12]. As most of the jobs of a secretary are of a routine nature, they can be handled by an automatic secretary. Of course, a human secretary and an automatic secretary can also work together to achieve better performance. There are a number of advantages using an automatic secretary instead of a traditional secretary. They are

1. Round-the-clock services
2. Less human errors
3. Less operating costs
4. Less communication problems

Motivated by some virtual secretary systems, this project develops an automatic secretary called ASSIST (Automatic Secretary System using Internet and Smart-agent Technologies). ASSIST can reduce the workload of a personal assistant or secretary by sharing the time consuming works. The works include meeting scheduling, meeting remaindering, email filtering, document handling, and

personal schedule keeping. As ASSIST is web-based, users can access the system anywhere through the HTTP protocol via a web browser or a WAP-enabled device.

In the existing virtual secretary projects, most of them support the scheduling of meetings. To organize a meeting effectively, different meeting scheduling mechanisms have been proposed. In most of the existing scheduling mechanisms, an arbitrary number of time slots are often used for scheduling purposes. A major contribution of this thesis is to propose a mathematical model and a Markov decision-based scheduling mechanism for computing the number of time slots. In addition, some matching algorithms are investigated to find the common time slot for a meeting.

For completeness, ASSIST also provides other services besides the meeting scheduling function. The extra functions are email function, document management, online address book service and meeting reminder service. Essentially these are the major tasks handled by a secretary.

1.2 Scope & Objectives

The aim of this project is to design an automatic secretary system capable of providing fundamental secretarial services. The focus is on meeting scheduling. As discussed later, our meeting scheduling algorithms can be used to find the optimal number of proposed time slots for scheduling a meeting. We have also

formulated a meeting scheduling problem by using a mathematical model and a Markov decision model.

In the mathematical model of the meeting scheduling problem, we assume that the scheduling process can continue until a common time slot is found. The Markov decision model considers the scheduling deadline. Also, the number of proposed time slots can be increased dynamically when approaching the scheduling deadline.

In addition, we have proposed different matching algorithms to search for the common time slot for a meeting.

In summary, the objectives of the research project are to:

1. design an automatic secretary (ASSIST)
2. develop a prototype to demonstrate some of its functions
3. develop a meeting scheduling mechanism
4. investigate some time slot matching algorithms for meeting scheduling purposes

1.3 Organization

The following is the organization of the remaining chapters of the thesis.

Chapter 2: Background study

It gives an overview of the existing virtual secretary projects, different meeting scheduling algorithms. It also describes the agent technology and the advantages of using agents.

Chapter 3: ASSIST

ASSIST (Automatic Secretary System using Internet and Smart-agent Technologies) is our proposed automatic secretary which provides basic secretarial functions, such as document management, email management and meeting scheduling service. In this chapter, the architecture, functions and the meeting scheduling protocol of ASSIST are described.

Chapter 4: A basic mathematical model for calculating the number of time slots for scheduling a meeting

This chapter presents a basic mathematical model for meeting scheduling. The objective is to find the optimal number of proposed time slots that can provide a minimum scheduling cost.

Chapter 5: A Markov decision model for calculating the number of time slots for scheduling a meeting

As an extension to chapter 4, we formulate a Markov decision model for computing the optimal number of time slots for scheduling a meeting in this chapter. The backward induction method for computation is used. Analytical and results are presented and discussed.

Chapter 6: Matching algorithm

After discussing the methods for finding the optimal number of proposed time slots in chapter 4 and 5, we investigate five matching algorithms for finding a common time slot. Simulation results are presented to compare these algorithms.

Chapter 7: Conclusion

We conclude the thesis in this chapter.

CHAPTER 2
BACKGROUND STUDY

In this chapter, we give an overview of the existing virtual secretary projects and the meeting scheduling projects, as well as mobile agent technologies.

2.1 Related Work on Virtual Secretary Projects

Hong Kong is an international trading center, where many professionals from all over the world come to do business. These professionals spend much time checking and replying to thousand of email messages, attending meetings and answering phone calls. Due to the rapid growth of Internet technology, professionals can make good use of the Electronic Meeting System (EMS) [3], [6], [9], [15], [31]. In order to reduce their workload, professionals should have an assistant. The job duties of these assistants are to organize meetings and make sure meetings do not overlap, filter out unimportant phone calls and email and remind their bosses to attend meetings. In recent years, a number of intelligent telephony systems [24], [28], [29] and virtual secretary systems (introduced below) have been developed. Motivated by existing electronic secretaries, we have developed a virtual secretary (ASSIST) to assist the

secretary for handling simple secretarial tasks. The details of ASSIST are given in Chapter 3. The advantages of using a virtual secretary are as follows:

- It can be accessed from anywhere, with no restrictions on place and time.
- It eliminates human errors.
- It provides around-the-clock service.
- It reduces operating costs.

In recent years, a number of virtual or electronic secretary systems have been developed by various universities and research institutions. Most of the systems support basic secretarial functions such as scheduling meetings, filtering email and managing documents. Three major automatic secretary projects are introduced as follows:

1. A secretary agent system based on HTTP client-server protocol

   With the aim of scheduling meetings effectively, Park and Kwak proposed a secretary agent system in [32]. Based on the scheduled data from the system's database, the secretary agent searches for the appropriate time slot(s) by using some predefined rules. The system architecture of the scheduling system is shown in Figure 1.

System architecture

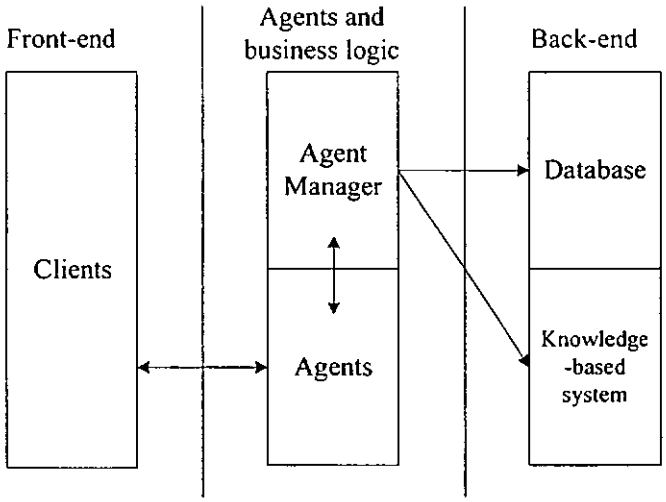

Figure 1. The system architecture of the software secretary proposed by Park and Kwak

As shown in Figure 1, the scheduling system proposed by Park and Kwak can be divided into three parts. The first part is the front-end interface. A user requests for a new meeting through the front-end interface. After the system selects the proposed time slots or confirms the meeting time, it returns the result to the user also through the front-end interface. The middle layer includes the agent manager and agents. The agent manager is stored in the agent manager server, while the other agents are located in the agent server. The agent manager is responsible for handling the requests, such as assigns the time slot to the meeting and chooses a set of proposed time slots. While the agent in the agent server is responsible for communicating with the user, transferring the request from the user to the secretary agent, as well as carrying back the result to the user. The last part is the back-end system, including the database and the knowledge-based database. The database is used to store the personal information of the users, their schedules, details of the proposed meetings and the confirmed meetings. While the knowledge-based database is used to store the facts for generating the rules when schedule a meeting.

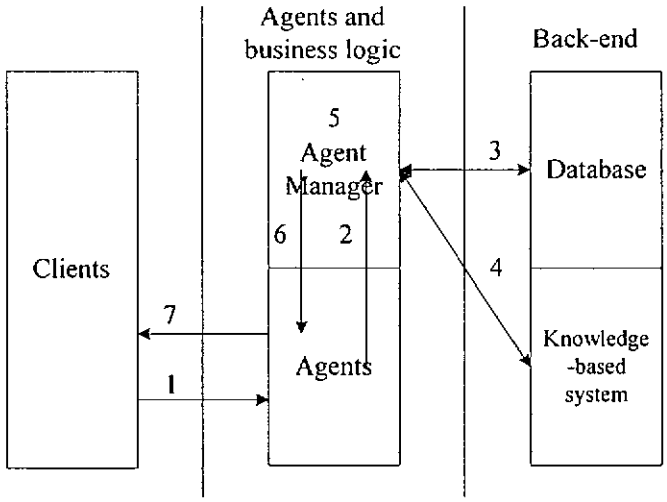

Figure 2. Data flow for handling a request

Figure 2 shows the algorithm of a request handling when a user submits a request. A host sends a new meeting request through the front-end interface to the scheduling system. The agent is created in the agent server and then receives the request through the HTTP protocol from the client side. After the agent receives and determines the type of requests, it sends the request to the agent manager in the agent manager server. The agent manager gets the personal schedule of the user from the database at the same time. It also gets the rules for selecting the proposed time slot from the knowledge-based database. After the secretary agent has chosen the appropriate time slot for the meeting, the agent obtains the result and then returns the proposed time slots to the user through the HTTP protocol such that the result can be displayed on the web browser. After the host and the meeting invitees confirm the meeting time, the secretary agent updates the user's schedule.

2. The virtual secretary project

Hartvigsen and other researchers proposed a virtual secretary (ViSe) system in [4], [5], [11], [12], [17] - [19]. It provides some basic secretarial functions, such as

managing files and filtering emails. ViSe and ViSe2 represent two different phases of the project. ViSe focuses on the single agent system whereas ViSe2 addresses the multi-agent architecture. Figure 3 and Table 1 below show the functional structure of the ViSe.

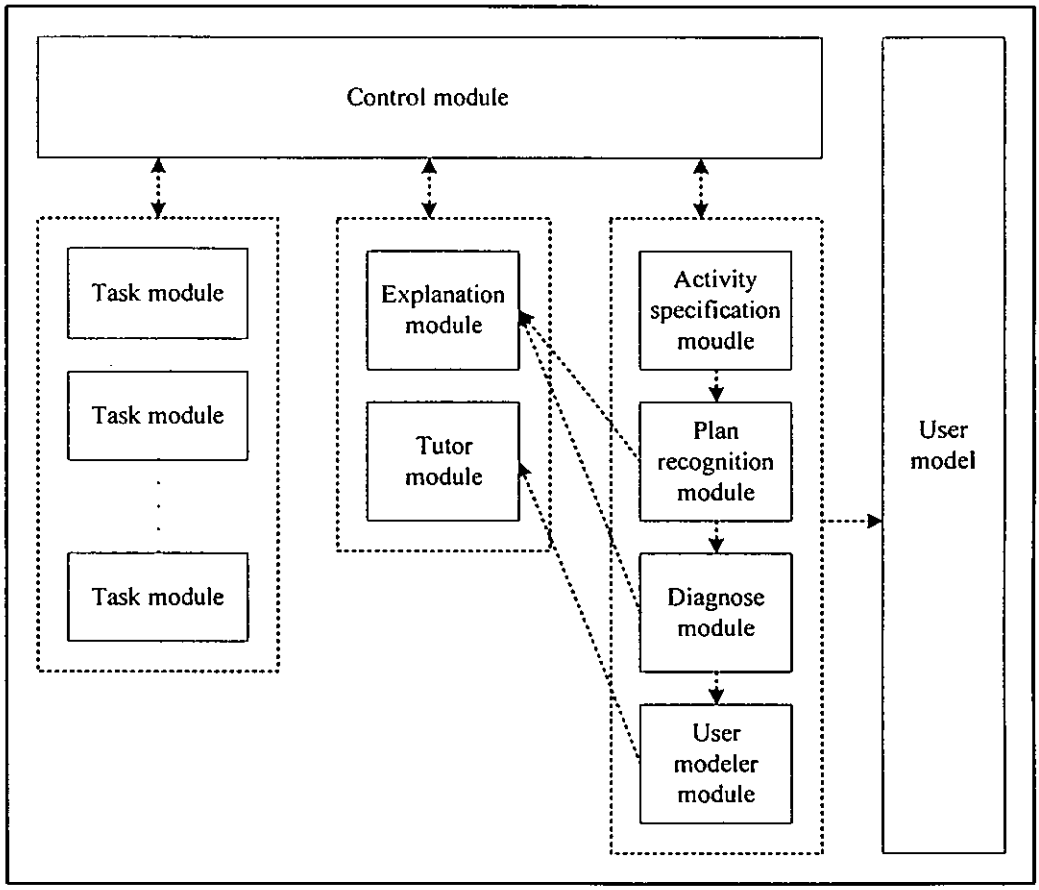

Figure 3. System modules of the ViSe

Table 1. Detailed functions of the modules in ViSe

Module	Function
Task module	This module is used to provide the secretary services for the users. There are more than one task module if ViSe provides multiple secretary functions, such as file retrieval, email

	filtering and personal environment setup.
Tutor module	This module is used to determine whether the ViSe user is a new user or not.
Explanation module	The explanation module is like a tutorial guide. It provides the user guide for the ViSe's users when the users seek for help or the system finds the user performing an illegal action.
Control module	The control module function likes the receptionist of the virtual secretary system, because it handles all I/O requests from the users and the task modules. In other words, it receives the task request from users. It also returns the result to the user after ViSe has finished the task.
Activity specification module	The user can specific a particular task and assign it to the ViSe through the activity specification module. Then the activity specification module follows the instruction to perform the required tasks.
Plan recognition module	This module is used to determine, recognize and record the user activity behavior. The activity behavior of the user forms the user model. The user module is stored in the system and is used by other modules, such as the tutor and explanation modules.
Diagnose	This module is used to determine and uncover the

module	misunderstanding and misinterpretation based on the user activity patterns recognized by the pattern recognition module. Action is passed to the explanation module and acts as an example of a wrong action.
User modular module	This module is used to update the user module based on the information provided by the activity specification module, plan recognition module and diagnose module.

The virtual secretary system proposed by Hartvigsen and other researchers can be divided into two phases as we mentioned before. The first phase is the ViSe. The system focuses on the use of the user model and solves the security issue for performing simple secretary tasks. The idea of the virtual secretary is to delegate the agent to perform the simple secretary tasks on behalf of the users. This is done based on the activity behavior (user model) of the users. Based on the mobile ability of the agents, they can travel from one site to another site to perform the task for the user. However, a security problem arises when the agents travels to a remote site. As a result, Hartvigsen and other researchers proposed that the agents carry the user model to the remote site only. In other words, no executable program is brought along with the mobile agents. As a result, the executable agent in the remote site uses the user model to understand the activity behavior of the user, and then perform the task accordingly. So the agent system should be pre-installed in the remote site. Figures 4 and 5 show how the system handles the request from the client and how the user model is transferred to the remote site.

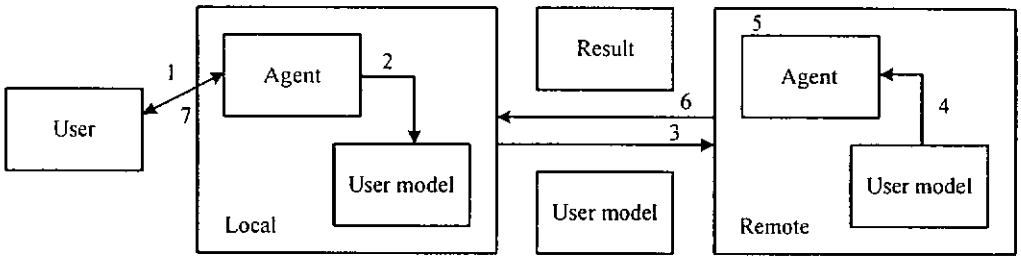

Figure 4. Data flow when ViSe handles a request

1. A user sends a request to the agent through the interface provide by the system.
2. The agent receives the request and gets the user model of the user.
3. The user model is cloned once and then transferred to the remote site.
4. The remote agent gets the cloned user model.
5. The remote agent uses the cloned user model to perform the task.
6. After completion, the result is sent back from the remote site to the local site.
7. The agent shows the result to the user.

Figure 5. Detailed protocol of request handling

Besides the security issue, there is also a network bandwidth problem. In order to reduce the size of the transferred data, only the user model is transferred to the remote site.

The ViSe implements the simple secretarial tasks, including the file retrieval, email filtering and the personal environment setting. The agent can help the user to retrieve the file from the remote site, even the user forgets the filename and the location of the file. The user model is copied and transferred from the local site to

the remote sites. Then the remote agents use the user model of the user and then form the list of the potential files from the remote sites. As a result, less bandwidth is used because no extra file is transferred from the remote site to the local site. In addition, the user model can keep the junk mail information and the desktop interface of the user. The agent uses the user model to filter the junk mails and set up the same interface when the user uses another computer.

3. Software secretaries: learning and negotiating personal assistants for the daily office work

Bocionek proposed a software secretary that serves as a personal assistant for negotiating and learning in [7]. The software secretary provides two main services: meeting arrangement and meeting room management.

System architecture of meeting arrangement

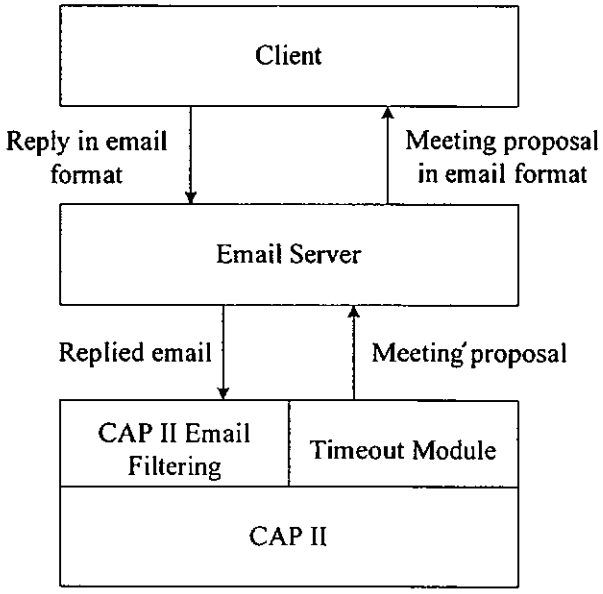

Figure 6. The simplified system architecture of the software secretary

Figure 6 shows the system architecture of the software secretary proposed by Bocionek briefly. Users request or respond to the system through email. CAP

(calendar apprentice) and CAPII (calendar apprentice II) are the central brains for scheduling a meeting, because they can give advices to the users on how to schedule and set up a meeting. Based on some algorithms, they are able to learn from the meeting arrangement history of the users, in order to provide the professional advices and comments for scheduling a new meeting. Users can accept or override the suggestion made by CAP or CAPII.

Besides the learning capability, CAPII can interact with the other CAP II's users or non-CAP II's users by means of electronic mail. When a user requests a meeting and confirms the desired meeting time, CAP II generates a proposal for the meeting and then sends it to all meeting invitees by email. The format of the email is predefined so it is readable by CAP II and the human users. As a result, both CAP II's users and non-CAP II's users can read the email and then reply to the host of the meeting accordingly. The CAP II identifies the emails by using keywords. Non-CAP II's users should reply the mail with in the predefined format. Otherwise, CAP II cannot understand the email content. For example, the subject of the email should be ended with the word "CAP-Request" which is followed by an event identifier. The event identifier is used to indicate that the email is associated with the meeting. Note that CAP II uses the word "CAP-Request" to recognize the email.

The system implements the timeout measurement in order to make sure that the replied mail is received. If CAP II does not receive the replied email from the invitees within a certain period, it sends the proposal to them again. The timeout module is used to alert the meeting invitees to resend the replied emails, non-CAP II users resend the email in the correct format.

The system addresses not only meeting scheduling but also room management. The room reservation apprentice (RAP) is used to handle the room reservation requested by departments in an organization. RAP is a knowledge-based system, similar to CAP II. It learns the vocabulary from the history of the requests, and then generates the rules for booking the rooms.

2.2 Related Work on Meeting Scheduling

As mentioned in the previous section, most virtual secretaries provide a meeting scheduling function [32]. Scheduling meetings is a major task of a secretary. Most secretaries should assist their supervisors in making appointments and scheduling meetings, contacting all invitees and confirming meetings. Scheduling meetings manually is tedious and time-consuming. As a result, a meeting scheduling module is implemented in the virtual secretary system. The meeting scheduling system can reduce the workload of a secretary by automatically searching for a free time slot for a meeting and contacting the meeting invitees via electronic mail.

Many academic and commercial parties suggest different meeting scheduling systems with different algorithms [8], [16], [21]. In the following section, some meeting scheduling projects are introduced.

1. Developing an Automated Distributed Meeting Scheduler

Sandip Sen has developed a distributed decentralized meeting scheduler [20], [35] - [38]. The system can be divided into six parts, shown in Figure 7.

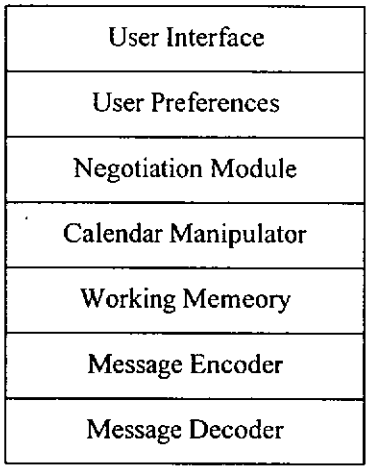

Figure 7. System components of the scheduling system

a. User interface

- User interface is used to accept the inputs from users.

b. User preferences

- The user preference module is used to store the user's preferences for scheduling a meeting.

c. Negotiation module

- When scheduling a meeting, agents should negotiate with each other, in order to find a common time slot. This process is done by the negotiation module.

d. Calendar manipulator

- The calendar of each user is kept in the calendar manipulator.

e. Working memory

- Working memory is a working area for agents.

f. Message constructor and decoder

- The message constructor is used to construct and send email to users.

- The message decoder is used to analyse the received emails.

A user sends a request to the system with the basic meeting information when he/she wants to hold a meeting. He/she becomes a host of the meeting and should specify the invitees, duration of the meeting, priority, a set of possible starting periods and a scheduling deadline of the meeting. The host announces the possible time slot(s) to the meeting invitees. The invitees check their schedules and return the replies, called bids. Invitees can propose another time slot(s), based on their schedules. Afterward, the host finds a common slot within the bids and sends the confirmation to the invitees. If all invitees can attend the meeting, then they mark their calendars as "occupied". However, if no common time slot is found, the host checks his/her availability on the time slot(s) proposed by invitees. The host sends the available time slot(s) to invitees and waits for their replies. The invitees send the bids to the host and then the host checks the common free time slot again. The system repeats the steps until the meeting can be scheduled.

The system provides a high level of privacy for the users, because each user has a private calendar. However, a tradeoff exists between privacy and total time for scheduling a meeting. If much information is transferred from the invitees to the

host, then the system becomes a centralized meeting scheduling. However, the system can assign a time slot to a meeting faster if it has a great deal of available information of the invitees.

In the above meeting scheduling algorithm, the host chooses the earliest common time slot for a meeting if more than one slot suits the meeting. In addition, Sen has proposed three strategies for choosing the best time slot.

a. Linear-early strategy

   - Using the earliest available time slot for the requested meeting is effective for small agent groups working on short calendars.

b. Linear-least-dense strategy

   - The linear-least dense strategy proposes to schedule a meeting in an evenly distributed manner.

c. Hierarchical strategy

   - The host selects the common time slot starting from the desirable week, then the day and finally the desirable time. This method is good for large organizations.

2. Privacy/Efficiency Tradeoffs in Distributed Meeting Scheduling by Constraint-Based Agents

Freuder, Minca and Wallace have proposed an agent-based meeting scheduling system and discussed the tradeoffs between privacy and efficiency in [14].

The agent-based meeting scheduling system is similar to Sen's project, which is mentioned in [20], [35] - [38]. The meeting scheduling system proposed by Freuder, Minca and Wallace allows invitees to provide the reasons why they are unavailable in the proposed time slot(s). In addition, invitees can provide extra unavailable time slots outside those proposed by the host.

3. PASHA – Personal Assistant for Scheduling Appointments

Schmeier and Schuperta have proposed a meeting-scheduling project with four scheduling meeting strategies in [34] This is called PASHA (Personal Assistant for Scheduling Appointments.)

a. Optimistic strategy

- This is the basic strategy for scheduling a meeting. The meeting host sends a request to all invitees with a proposed time slot. If some invitees cannot attend the meeting, then scheduling has failed. No rescheduling is required.

b. Deliberate strategy

- This is an extension of the optimistic strategy. The host of the meeting sends a request to all meeting invitees with a proposed time slot. If the meeting cannot be scheduled, then the host repeats sending other time slots to the invitees until the meeting can be scheduled.

c. Industrial strategy

- The person who caused the failure of scheduling is responsible for proposing another time slot for the meeting.

d. Complex strategy

- The rescheduling of a scheduled meeting is allowed in the complex scheduling strategy.

4. A Meeting Scheduling System for Global Events on the Internet

The project named "A Meeting Scheduling System for Global Events on the Internet" [1] introduces how to handle the meeting across different countries. The time difference is a stress factor. If the participants of the meeting come from different countries, they have different official working hours. Therefore, the meeting should be held when deviation from their working hours is minimum.

This project introduces the idea of a quorum. The meeting invitees are separated into several groups, and each group should have a minimum number of attendees. The meeting cannot be scheduled successfully if insufficient attendees are there. In addition, the system uses the standard of CALSCH WG to develop a calendar.

5. Agent-based Meeting Scheduling: A Design and Implementation

Jennings and Jackson have proposed an agent-based meeting scheduling system which is called "Agent-based Meeting Scheduling: A Design and Implementation" in [25]. The following diagram shows the architecture of the system.

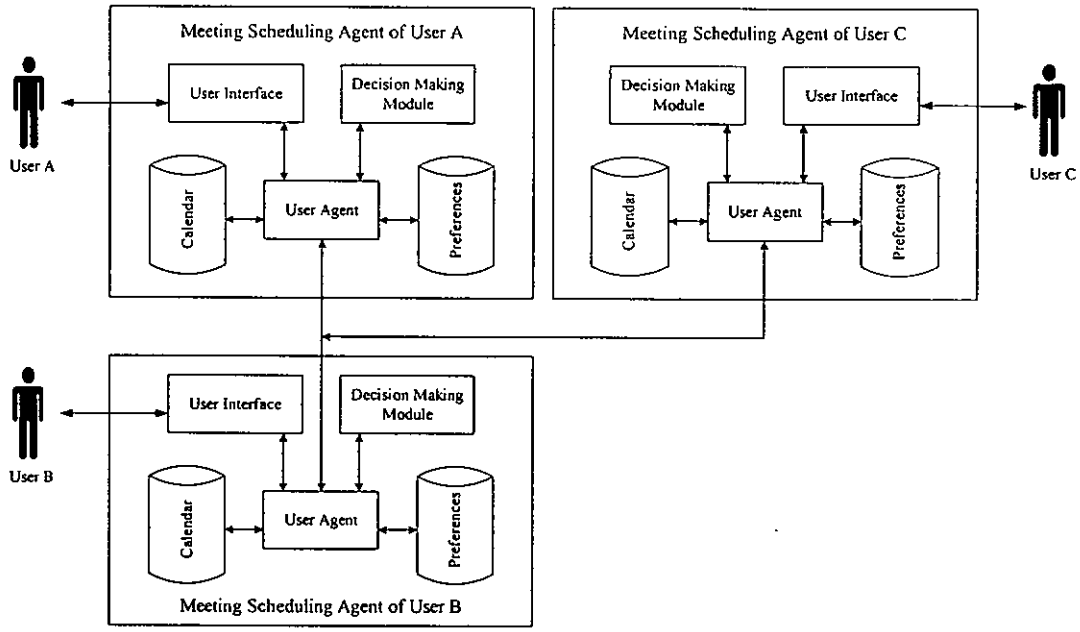

Figure 8. System architecture of the agent-based meeting scheduling system

As shown in Figure 8, users access the system through the user interface. Users should be able to request a meeting by submitting the basic information of the meeting to the system, through the user interface. Furthermore, users can communicate with their agents via the user interface. Agents of the user should be able to make decisions and negotiate with other agents. The host's agent uses the decision-making module to find a suitable time slot for the meeting, based on all proposed time slots from the invitees.

Each user has a private calendar, from which the user's agent checks the confirmed meetings. Only the user's agent can access the scheduled meeting. The user's preferences are stored in the private database, and the user's agent proposes the time slot(s) for a meeting based on the preferences. The agents within the system can communicate with each other by bids or by announcing the confirmation of a meeting. The agent shows the user's preferences for the proposed time slots by sending the bid to the system.

Meeting Scheduling Algorithm

When a user wants to have a meeting with the others, for example, User A wants to invite Users B and C to join the meeting, User A should send a request to the agent via the user interface provided by the system. Figure 9 below shows the protocol of a meeting scheduling.

1. The host of the meeting sends a request to his/her agent with the names of the invitees, purpose of the meeting, desired duration and constraints of the meeting.
2. After the host's agent receives the request, it sends a message to the agents of all invitees about the meeting.
3. The invitee's agent should determine the potential time slot(s) for the meeting, according to the constraints, duration of the meeting and preferences of the user.
4. The invitee's agent gives a rating on the potential time slot(s), based on the user's preferences.
5. The agent returns a bid with the potential time slot(s) and the corresponding rating.
6. The host's agent receives the bids from the agents of all invitees.
7. The host determines the common time slot which has the highest rating and is over the threshold of rating.
8. The host sends the common time slot to the agents of the invitees and then waits for the confirmation from these agents.
9. The invitees' agents return the confirmation bids.
10. If all invitees agree to attend the meeting at the common time slot, the host reconfirms the meeting time with all invitees.
11. If some invitees cannot attend the meeting, the host's agent repeats step 8.
12. If the host's agent cannot find a common time slot, then it repeats step 2.

Figure 9. Scheduling protocol of the system

The host of the meeting sends a request with the basic information of the meeting to the host's agent in step 1. The basic information includes duration and constraints of the meeting, and the names of the invitees. The constraints of the meeting include the threshold of the rating, etc. In steps 3 and 4, an invitee's agent chooses several proposed time slots for the meeting, based on the information about the meeting. It gives a rating on each proposed time slot based on the user's schedules and preferences, which are stored in the private calendar and the preference database. Then the host determines the common time slot for a meeting from the bids by the invitees, in step 7. The host's agent searches for a common time slot over the threshold and with the highest rating. If no common time slot is found, then the host's agent asks for further bids from the agents of the invitees.

The meeting scheduling system also allows rescheduling the scheduled meeting. Rescheduling occurs if the occupied time slot is the only common time slot of another meeting that is under scheduling. In addition, the rescheduling occurs if the scheduling meeting is more important than the scheduled meeting.

Special features of the functions & algorithm

We summarize the features of the meeting scheduling system proposed by Jennings and Jackson [25].

a. Decentralized system

The meeting scheduling system is a decentralized system, because each user has a calendar for keeping his/her schedules, as shown in Figure 2. The calendar can be accessed by the user's agent only. The decentralized system

provides a high level privacy for users, because only a user's agent knows the schedules of the user.

Furthermore, the meeting scheduling system works properly even if the user's agent does not work. The meeting scheduling system is independent of all agents. The working efficiency of the decentralized system is higher than that of the centralized system.

b. Initiative from invitees

Invitees propose the time slots of a meeting based on the constraints and duration of the meeting, their schedules and preferences, as shown in step 3 of Figure 9. Although the host's agent knows all the available time slots of the invitees, the host may not be available at the given time slots. The meeting scheduling system is a decentralized system, so the invitees propose the time slots based on their schedules instead of the host's schedule.

c. Rescheduling a confirmed meeting

Rescheduling a confirmed meeting occurs when the common time slot of a meeting of the invitees or the host is assigned to another scheduled meeting. The host's agent sends a rescheduling request to the host's agent of the confirmed meeting, and the host of the confirmed meeting searches for another common time slot for the meeting and reallocates it to the common time slot. As a result, the pre-occupied time slot can be released after the rescheduling of the confirmed meeting.

The advantage of rescheduling is to provide a free time slot for a meeting, so it can be scheduled successfully. However, rescheduling means that another, pre-scheduled, meeting must be rearranged into another time slot. Such a rescheduling raises costs. In addition, the rescheduling of a confirmed meeting causes a deadlock. When the scheduled meeting is rearranged to another time slot, it may cause another scheduled meeting to be rescheduled, if the common time slot is occupied by the scheduled meeting. As a result, rescheduling of a scheduled meeting may result in a chain effect on all scheduled meetings. To reduce the side effect of rescheduling, Jennings and Jackson proposed only important meetings should cause rescheduling of a confirmed meeting [25].

Rescheduling is a common function in meeting scheduling systems, i.e., the system can reassign a time slot to a scheduled meeting. In addition to the meeting scheduling project proposed by Jennings and Jackson in [25], Jeong, Yun and Jo proposed a meeting scheduling system with a rescheduling function in [26]. The following meeting scheduling system shows the detailed rescheduling algorithm in [26].

6. Cooperation in Multi-agent System for Meeting Scheduling

Jeong, Yun and Jo proposed an agent-based meeting scheduling system called "Cooperation in multi-agent system for meeting scheduling" [26]. The aim of the project is to resolve the conflict of a time slot with two or more meetings. The solution is to reschedule a scheduled meeting to another suitable time slot and use the extended cooperation. The details of the extended cooperation are given later. Figure 10 shows the system architecture of the meeting scheduling system

proposed by Jeong, Yun and Jo in [26]. Three different agents exist in the scheduling system, and they are responsible for different parts in the system. They are facilitator, personal assistant agent (PA) and meeting scheduling agent (MA).

The facilitator is responsible for keeping the address of the MA and all PAs. When a user creates a new PA, it should register with the facilitator so it can receive messages from the MA or other agents.

The PA is created by a user and is a personal agent of its owner. It is responsible for accessing the user's calendar, requesting a meeting on behalf of its user, keeping track of and updating the scheduled meetings, manipulating the user's private meetings, accessing the user's preferences and negotiating with the MA to find a time slot for a meeting. When the MA cannot find a common time slot for a meeting, the PA negotiates with the MA to move the private meeting of the user to another time slot, in order to free the time slot for a meeting.

The MA is responsible for organizing a meeting requested by a user's PA and acts as a coordinator to schedule a meeting. When it receives a scheduling request, it talks with the invitees' PAs to find a common time slot for a meeting. If no common time slot is found among the host and all meeting invitees, the MA should request the invitees' PAs to change the pre-scheduled private meeting to another time slot, in order to release a time slot for a requested meeting.

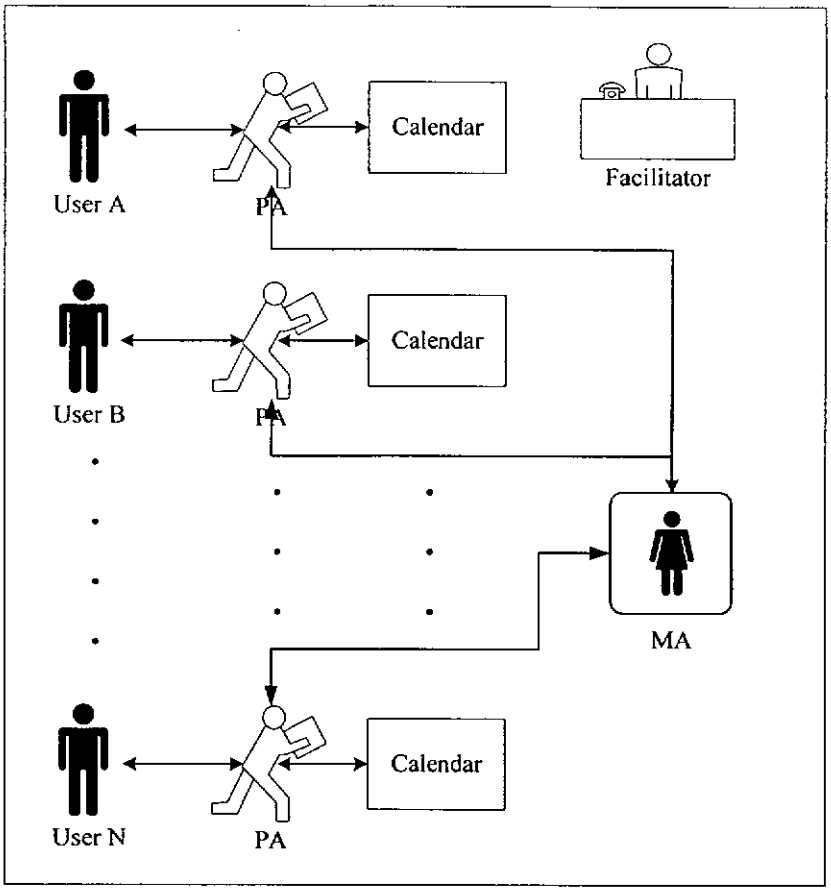

Figure 10. System architecture of the meeting scheduling system

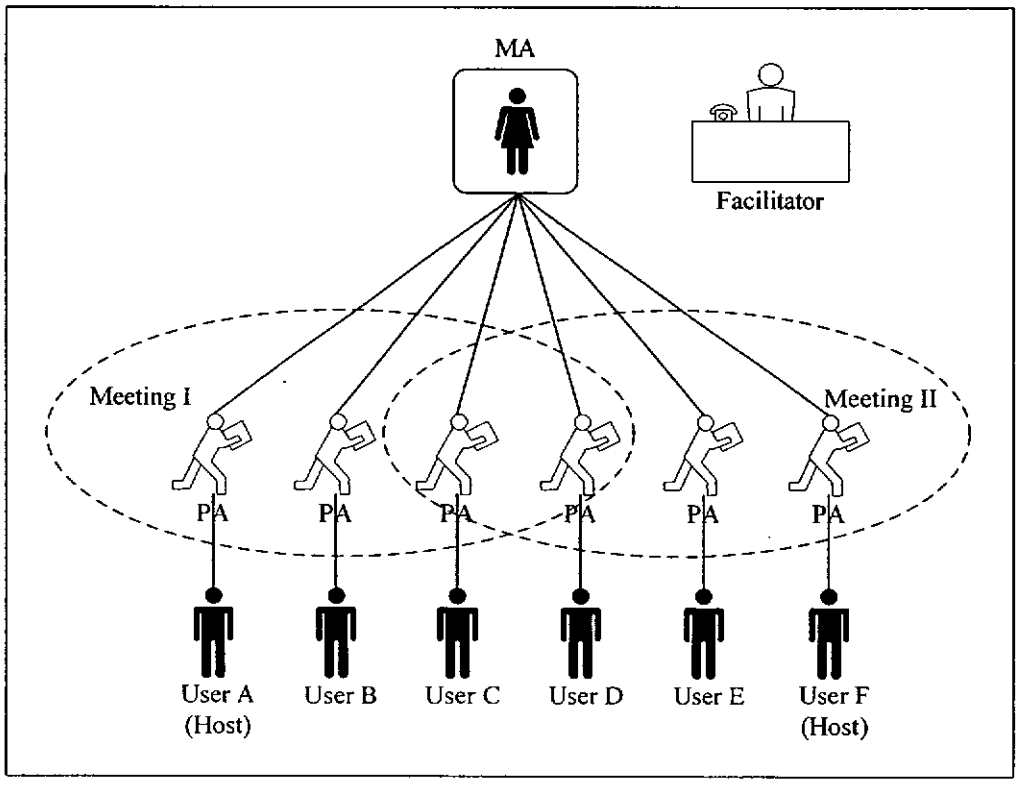

Figure 11. The relationship of facilitator, MA and PAs in a meeting scheduling system

As shown in Figure 11, there is one facilitator and one MA in the meeting scheduling system. The user creates one PA only and the PA represents the user. A user can be involved in more than one meeting at the same time, and then the PA negotiates with the MA to find time slots for the meetings.

Meeting Scheduling Algorithm

Figure 12 shows the summary of the scheduling protocol of the project proposed by Jeong, Yun and Jo in [26]. The whole meeting scheduling process can be divided into five phases: contract net protocol phase, cooperation phase, rescheduling phase, non-committed phase and committed phase. The detailed protocol of each stage is shown in Figures 13 to 17.

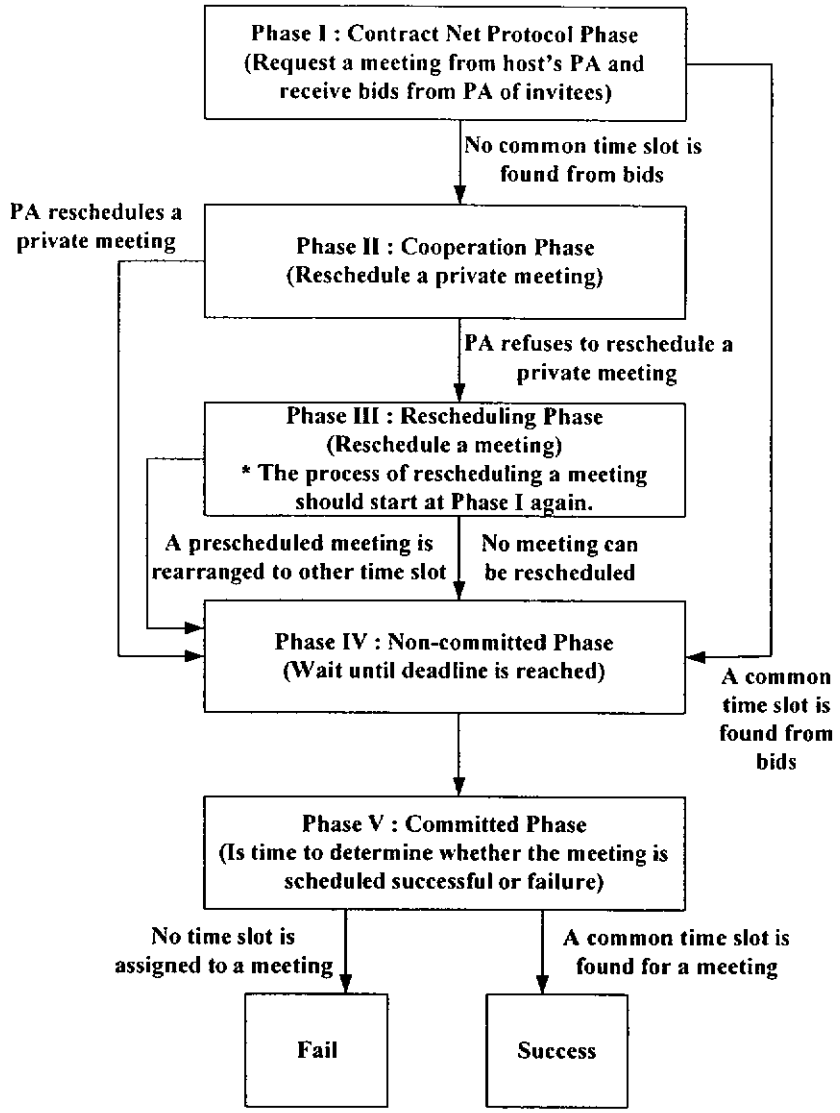

Figure 12. Summary of the meeting protocol

When a user wants to hold a meeting, he/she sends a request to the PA. The PA receives a request from the user and then sends a meeting request to the MA. The request should include the basic information of a meeting, such as duration, the name of the invitees, proposed meeting period and the end time of the scheduling. The MA announces to the PAs of all invitees the new requested meeting. Based on the proposed period of a meeting, the PAs respond to the MA with their preferences (high/medium/low) on the free time slot(s), the moving cost of each movable private meeting and the time slot of the non-movable private meetings within the period. The MA determines the time slot for a meeting if a common

time slot exists among all invitees and the end time of the scheduling is not reached. However, if no common time slot exists, the MA searches for any private meeting of the invitees that can be rescheduled to free a time slot for a meeting. The detailed protocol of the contract net protocol phase is shown in Figure 13.

Phase I: Contract Net Protocol Phase

1. A host sends a meeting request to his/her personal assistant agent (PA_{host}) with the meeting information, including the new meeting date, duration of the meeting and end time of the scheduling.
2. The PA_{host} sends a request to the MA about a new meeting, with the information.
3. The MA announces the information about the new meeting to the PAs of all invitees ($PA_{invitees}$).
4. $PA_{invitee}$ identifies the free time slots, moveable and unmovable scheduled meetings of its owner in the proposed meeting date.
5. $PA_{invitee}$ identifies the moving cost of the movable scheduled meetings.
6. $PA_{invitee}$ shows the preference, i.e., high, medium, low of each free time slot.
7. $PA_{invitee}$ replies to the MA containing the preference of each free time slot and the moving cost of each movable scheduled meeting.
8. The MA waits for the replies from $PA_{invitees}$.
9. The MA searches for a common free time slot for a meeting from the bids.
10. If more than one common free time slot is found, the MA chooses the one with the highest rating of the preferences.

Figure 13. Detailed protocol in the contract net protocol phase

The diagram below shows the detailed protocol in the cooperation phase. The objective of the cooperation phase is to rearrange a scheduled private meeting to another time slot, in order to free a time slot for a requested meeting. The cost of the rescheduling is a consideration factor when rescheduling is required. If the PA refuses to reschedule a private meeting, the MA searches for potential common time slot with a higher rescheduling cost. The detailed protocol is shown in Figure 14.

Phase II: Cooperation Phase

1. If no common time slot is found within all proposed time slots by $PA_{invitees}$, then:
2. The MA determines a potential free time slot if the $PA_{invitee}$ agrees to move the private scheduled meeting to another time slot, in order to release the potential free time slot.
3. The MA chooses the potential free time slot with the lowest moving cost.
4. The MA sends a rescheduling request to the corresponding $PA_{invitee}$.
5. If $PA_{invitee}$ changes the private meeting to other time slot and releases the time slot to the proposed meeting, then the meeting can be scheduled successfully.
6. Otherwise, the MA searches for another potential time slot with a higher rescheduling cost if the $PA_{invitee}$ moves the private meeting to another time slot.

Figure 14. Detailed protocol in cooperation phase

In the rescheduling phase as shown in Figure 15, the MA rearranges the scheduled meeting to another time slot, because the scheduled meeting occupies a time slot

which is suitable for a requested meeting. Rescheduling the meeting occurs only if the scheduled meeting should be in the non-committed phase and can be rescheduled to another time slot successfully. Furthermore, at least one of the attendees of the scheduled meeting is the invitee of the requested meeting.

Phase III: Rescheduling Phase

1. If the MA cannot find a common time slot after the cooperation phase, it searches a potential free time slot that needs rescheduling of a scheduled meeting.
2. The potential time slot should satisfy the following criteria:
   a. The scheduled meeting at the potential time slot should involve one invitee or more of the proposed meeting.
   b. The scheduled meeting should be in a non-committed phase.
   c. The scheduled meeting can successfully be rescheduled to another time slot.
3. The MA searches for another free time for the scheduled meeting.
4. The MA sends a rescheduling request to the $PA_{invitees}$ of the scheduled meeting.
5. If the scheduled meeting is changed to another time slot and frees the potential time slot, then the proposed meeting can be scheduled to the potential time slot.
6. The MA sends an announcement to all $PA_{invitees}$ to confirm the meeting time.

Figure 15. Detailed protocol in rescheduling phase

Phase IV: Non-committed Phase

1. The requested meeting is in a non-committed phase, if
    a It is before the end time of the scheduling.
    b It does not matter whether or not a common time slot is found.
2. The MA waits to change a time slot of a private meeting if the requested meeting schedule is a failure.
3. The MA attempts to move the scheduled meeting to another time slot, in order to free the time slot for the requested meeting.

Figure 16. Detailed protocol in non-committed phase

The protocol of the non-committed phase is shown in Figure 16. If a scheduled meeting is rearranged to another time slot, then the time slot can be free for the requested meeting. Although the meeting is scheduled to a particular time slot, the MA waits until the end time of the scheduling. This is in the non-committed phase. It is possible that the meeting is rescheduled to another time slot if the scheduling is in a non-committed phase, because the occupied time slot is suitable for another meeting.

Figure 17 shows the protocol of the committed phase. When the deadline of the scheduling is reached, the MA announces the meeting time to the host's PA and all invitees' PAs if a meeting is scheduled successfully. The MA reports the failure of scheduling to the PA of the host if no common time slot is assigned to the requested meeting.

Phase IV: Committed Phase

1. When the deadline of a requested meeting is reached, then:
   a. The MA sends a confirmation to the PA of the host and all $PA_{invitees}$ about the meeting time if the requested meeting is scheduled successfully.
   b. $PA_{invitee}$ updates the calendar of its user.
   c. MA reports to the host if the request is failure.

Figure 17. Detailed protocol in committed phase

Special features of the functions & algorithm

The meeting scheduling system proposed by Jeong, Yun and Jo in [26] is similar to the one proposed by Jennings and Jackson in [25]. Both are agent-based meeting scheduling systems and use the decentralized database for storing the calendar of the users.

a. Decentralized system

Each user has a calendar for storing scheduled meetings, including private and non-private meetings. Furthermore, it stores the user's preferences on the time slots. Only the user's PA can access the calendar of its user. As a result, the meeting scheduling system provides a high level of privacy for the users.

b. Rescheduling of a confirmed meeting

Rescheduling of a scheduled meeting is allowed in the meeting scheduling system proposed by Jeong, Yun and Jo in [26]. As explained in the previous meeting scheduling system, rescheduling allows a rearrangement of a

scheduled meeting to another time slot and frees the original time slot for a requested meeting. The scheduled meeting is reassigned to another time slot only when it is in the non-committed phase and a new common time slot is found for the rescheduled meeting.

c. Rescheduling a private meeting

In the cooperation phase, a private meeting is allowed to move to a new time slot. The detailed protocol is shown in Figure 14. If no common time slot is found for a requested meeting, the MA reschedules a private meeting of the invitee first, rather than a scheduled non-private meeting. The cost of rescheduling a private meeting is smaller than the rescheduling cost of the non-private meeting, because the movement of a private meeting affects the private meeting's owner only. The advantages of using the cooperation strategy rather than the rescheduling strategy are:

- It minimizes the rescheduling cost.
- Fewer participants are affected if rescheduling a private meeting.

d. Non-committed period for a requested meeting

No matter a whether common time slot is found for a meeting or not, after the contract net protocol phase, cooperation phase and rescheduling phase, the requested meeting should wait in the non-committed phase before the end time of the scheduling. The purposes of the non-committed period are to:

- Allow the rescheduling of a non-committed scheduled meeting to another time slot, in order to free the current time slot for a requested meeting.

- Check the new available time slot for the unscheduled meeting periodically. Because the invitee's PA may delete a private meeting or move a private meeting to another time slot and release the pre-occupied time slot, the PA should inform the MA. The released time slot may be suitable for the requested meeting.

The non-committed period enhances the flexibility of meeting scheduling. The scheduled meeting can be rescheduled to a time slot during a non-committed period, so the original time slot is released for the requested meeting. As a result, the requested meeting can be scheduled successfully. Furthermore, an unscheduled meeting can be assigned to a free time slot during the non-committed period when someone changes the private meeting to another time slot. The non-committed period is for an unscheduled meeting waiting for a common time slot.

e. Number of proposed time slots

The MA and the PA of invitees do not have to propose any time slot for a meeting and determine the number of proposed time slots. The number of time slots is determined by the host of a meeting. When the host's PA requests a meeting from the MA, it should send the proposed period of a meeting. Every PA of the invitees should show the preference on each time slot within the proposed period.

If the host proposes a long period for a meeting, the invitee's PA should give the preference on each time slot within the proposed meeting period, one by one. The MA receives the bids with users' preferences from all invitees, and

then spends time determining the free common time slot. When the host proposes a longer meeting period, the MA should spend time checking the common time slot among the invitees. In addition, the invitee's PA spends more time filling in the preference of the user in each time slot.

However, if more time slots are required to fill in the rating by the PA, the invitee loses privacy. The PA releases the meeting time of the private meeting and the rating of the time slots within the proposed meeting range. However, the MA can find free common time slot more easily if the host proposes more time slots.

7. Multi-Agent Based Decision Mechanism for Distributed Meeting Scheduling System

Ashir, Joo, Kinoshita and Shiratori proposed a meeting scheduling system in [2] which does not provide a rescheduling function, not like the previous two meeting scheduling projects [25], [26]. They proposed the idea of a quorum, discussed later. The system architecture of the meeting scheduling system is shown in Figure 18.

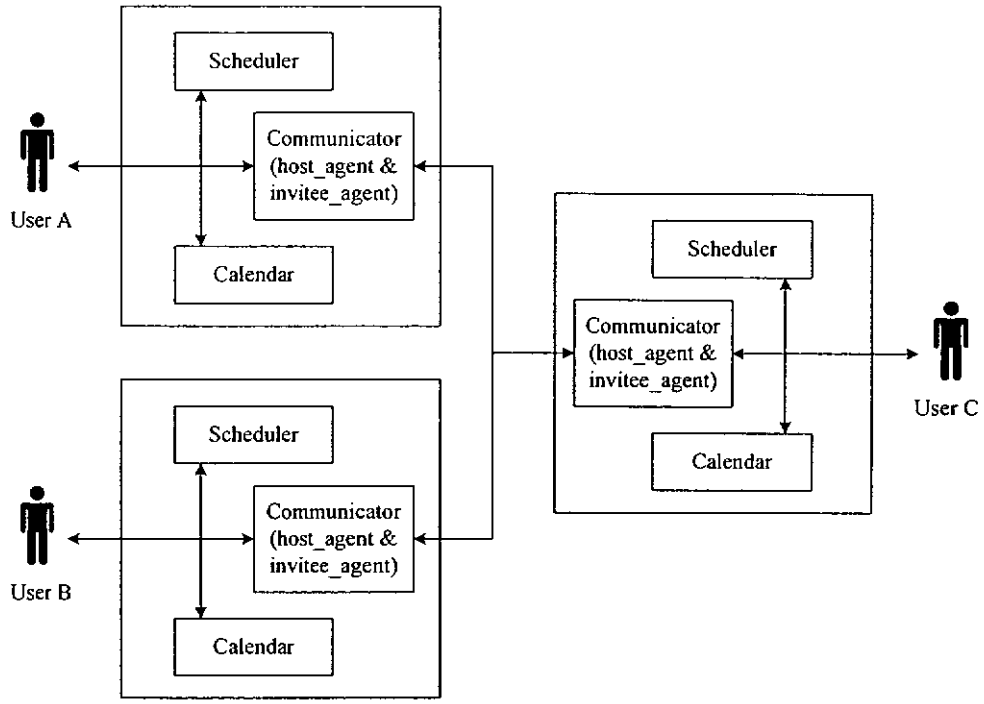

Figure 18. System architecture of the meeting scheduling system

The system architecture of the meeting scheduling is similar to the meeting scheduling system proposed by Jennings and Jackson [25]. As shown in Figure 18, the user consists of a communicator, scheduler and calendar. The following table shows the responsibility of the communicator, scheduler and calendar.

Table 2. Responsibility of scheduler, communicator and calendar

Agent Name	Responsibilities
Communicator (host_agent)	The communicator can be divided into a host_agent and an invitee_agent. Both agents are used to communicate with other agents. If a user wants to call a meeting, the user sends a request to a host_agent. Then the host_agent forwards the request to scheduler with the basic information of the meeting.
Communicator (invitee_agent)	The invitee_agent is one of the communicators. It receives an announcement about the new meeting from the scheduler and then

	sends the bid back to the scheduler.
Scheduler	The scheduler acts as a centre of the meeting scheduling system, because the scheduler is responsible for processing the meeting scheduling request and determining the best common time slot for a meeting.
Calendar	The calendar keeps and accesses the schedules of its user. There are three statuses of the time slot: busy, pending or empty status. If a time slot has already been assigned to a meeting, it should be in a busy status, and no meeting can be assigned to the time slot. A time slot in an empty status means no meeting is assigned to the time slot. A scheduler can assign the time slot in the empty status to a requested meeting if the time slot is suitable for a meeting. A time slot in a pending status means a meeting may occupy the time slot. If the scheduler decides to assign the time slot to a meeting, then the status of the time slot is changed from pending to busy. Otherwise, the status of the time slot is changed from pending to empty if the scheduler agent releases the time slot for other meetings.

Meeting Scheduling Algorithm

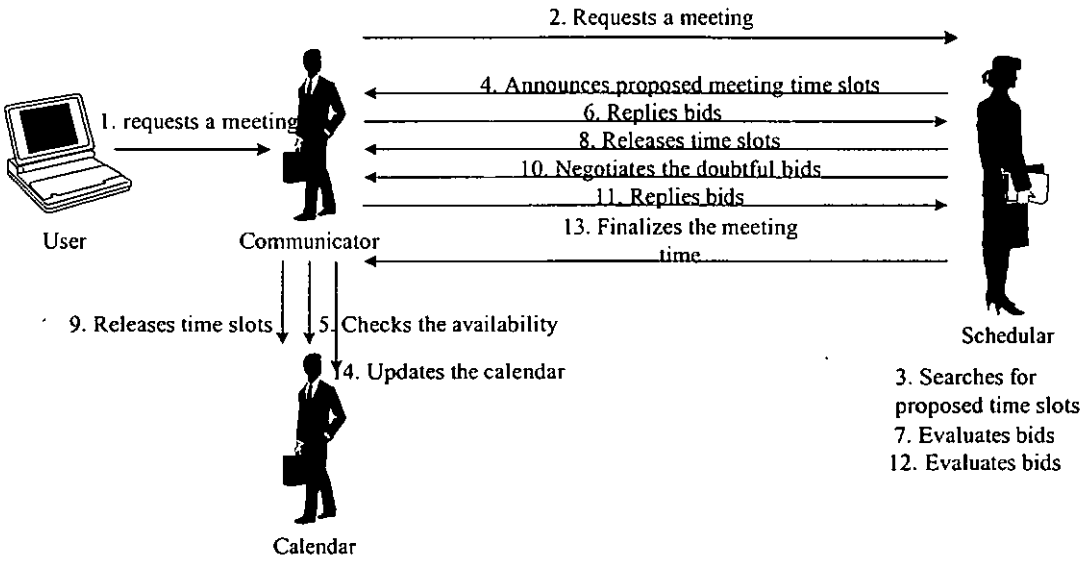

Figure 19. The workflow of the scheduling system

Figure 19 shows the whole meeting scheduling protocol proposed by Ashir, Joo, Kinoshita and Shiratori in [2]. As shown in the figure, the meeting scheduling process can be divided into six stages. They are generative, announcing, receive_bid, evaluating, negotiation and the awarding stage, represented in Figures 20 to 25 respectively. Simply put, when the user wants to have a meeting with the others, he/she sends a request to the host_agent. The host_agent searches all possible time slots in the generative stage. The host_agent informs the invitee_agents in the announcing stage. Then the invitee_agent checks the availability and returns the bid to the host_agent in the receive_bid stage. After the receive_bid stage, the host_agent receives all bids and then searches for a common time slot in the evaluating stage. If no common time slot is found, then the host_agent sends the message to the invitees and persuades them to attend the meeting at the chosen time slot in the negotiation stage. Finally, the host_agent tells all invitees the time of the meeting in the awarding stage.

Figures 20 to 25 show that the scheduling protocol proposed in [2]. The protocol can be divided into 6 stages, they are generative stage, announcing stage, receive_bid stage, evaluating stage, negotiation stage and awarding stage.

Stage I: Generative stage

1. The host of the meeting sends a meeting request to the host_agent about the new meeting. The request information includes the topic of the meeting, duration, proposed time range of the meeting and quorum.
2. The host_agent receives a request and then forwards it to a scheduler.
3. The scheduler depends on the duration of the meeting and the proposed time range to determine all possible time slots for a meeting. The algorithm for finding all possible time slots within the proposed time range is discussed in Figure 26.

Figure 20. Protocol in the generative state

Stage II: Announcing stage

1. The scheduler announces to all invitees about the requested meeting, including the topic, duration and all proposed time slots and who is the host_agent.

Figure 21. Protocol in the announcing state

Stage III: Recevie_bid stage

1. The invitee_agent receives the meeting request from the scheduler and then checks the availability of each proposed time slot.
2. The invitee_agent checks the proposed time slot is in busy, pending or empty status. The status of the time slots is used to indicate whether the invitees can attend the meeting at the proposed time slot. The invitee_agent returns the bid to the host_agent. If the invitee can attend the meeting, the invitee_agent sends back the bid with a "yes" answer. However, the invitee_agent sends the negative bid with "no" or "doubtful" if the invitee is unable to attend or is unsure.
3. The invitee_agent changes the time slots proposed by the scheduler from empty to pending if returned bids at the proposed time slot are yes or doubtful.

Figure 22. Protocol in the receive_bid state

Stage IV: Evaluating stage

1. The host_agent receives all bids from the invitee_agents and then forwards them to the scheduler.
2. The scheduler receives and then evaluates the bids.
3. The scheduler finds all unsuitable time slots, i.e., the time slot cannot be assigned to the meeting because it cannot meet the requirements of the meeting.
4. The host_agent sends a released notice to the invitee_agents who send the bid at the unsuitable time slot with a yes or doubtful answer. Then the host_agent tells them to change the status of the unsuitable time slots from pending to empty.
5. The host_agent finds the common time slot among the bids. The common time slot should be satisfied the quorum requirement. If a common time slot is found, then the host_agent enters the awarding stage. If no suitable time slot is found, then scheduler negotiates with the invitee_agents.

Figure 23. Protocol in the evaluating state

Stage V: Negotiation stage

1. In the Negotiation stage, if no common time slot is found for the meeting, the scheduler negotiates with the invitees in the compulsory group first.
2. The scheduler sends a message to the invitee_agents in the compulsory group who send a doubtful bid. The scheduler sends along with the requirement on the quorum to the invitee_agents.
3. After the invitee_agents return the positive bids to the scheduler, the scheduler sends a message to the invitee_agents who send a doubtful bid and persuades them to change the bid from doubtful to yes.
4. The scheduler receives the bids from the invitee_agents and evaluates the bids again. If the result satisfies the quorum of each group, then the time slot is assigned to the meeting.

Figure 24. Protocol in the negotiate state

Stage VI: Awarding stage

1. The scheduler informs the invitee_agents of the confirmed meeting time.
2. The invitee_agent changes the status of the confirmed time slot in the calendar from pending to busy. Also, it releases the unused time slots and changes their status from pending to empty.

Figure 25. Protocol in the award state

Method of finding all possible time slots

Equations (1) and (2) are used to calculate the number of all proposed time slots within a particular period, in which

- D_{min}: difference between start time and end time (in minutes)
- S: number of time slots
- T_e: start time of the time range
- T_s: end time of the time range
- d: duration of a meeting (in minutes)
- K: number of mini-slot in an hour

$$D_{min} = T_e - T_s \qquad (1)$$

$$S = (D_{min} - d)/(K+1) \qquad (2)$$

For example, a host wants to hold a meeting between 2:00 p.m. and 4:00 p.m. Each mini-slot is equal to 10 minutes and the meeting lasts for 30 minutes. As a result, the number of time slots for the meeting is equal to12, as shown in Figure 26.

T_e = 4:00p.m.

T_s = 2:00p.m.

According to equal 1, then D_{min} = 120 (in minutes)

d = 30

K = 6

Therefore,

$S = (120 - 30)/(6+1)$

Figure 26. Example on calculating the number of time slots

Special Features of the functions & algorithm

The meeting scheduling project is similar to the previous projects that are proposed by Jeong, Yun and Jo in [26] and Jennings and Jackson in [25]. All systems are agent-based scheduling systems and implement a decentralized database. Each user has an agent to organize and access his/her schedules. Furthermore, one of the agents is assigned to schedule a meeting of all invitees and the host.

a. Decentralized system

The meeting scheduling system proposed by Ashir, Joo, Kinoshita and Shiratori [2] uses a decentralized database. No one can access the schedules except the user's agent. The decentralized system can protect the privacy of the user.

b. An appointed agent for scheduling

The scheduler agent is assigned to find the common time slot for a requested meeting. This is the centralized scheduling agent approach. The distributed scheduling method can reduce the loading on a single centralized scheduling agent. Furthermore, if the dedicated agent does not work properly, the whole scheduling system cannot find a common time slot for a meeting, because only one agent is responsible for scheduling.

However, the number of agents increases in the decentralized scheduling agent approach. The number of scheduling agents is proportional to the number of users. If more agents are created, more resources are needed. Furthermore, some scheduling agents are idle if the users request no meeting. They consume resources even when they are idle, so they waste the resources.

c. Doubtful bid

When an invitee_agent returns a bid to the scheduler, it can answer "yes", "no" or "doubtful" for a particular time slot. The invitee_agent returns the positive bid with "yes" if the invitee is available at the time slot. The invitee_agent returns a negative bid with "doubtful" if the time slot is in the pending or empty status. The time slot is in the pending status if it is locked and may be assigned to another meeting. The invitee_agent sends the bid with "doubtful" if the time slot is in the empty status, because the attendance of the meeting is under consideration.

The purpose of a doubtful bid is for negotiation. When the scheduler cannot find a common free time slot among the bids from all invitees, then it tries to

persuade the doubtful bid owners to attend the meeting. If they change the doubtful bid to positive, then the attendance of the group satisfies the quorum requirement. As a result, the meeting can successfully be held at the time slot. The implementation of a doubtful bid can enhance the scheduling system to easily find a common time slot. The meeting can be assigned to the time slot after the negotiation with the owner of the doubtful bid.

The purpose of the doubtful bid is similar to the rescheduling function, because both methods involve rescheduling the confirmed meeting. However, rescheduling must affect the scheduled meeting and the corresponding host and invitees, but if the invitee_agent changes the doubtful bid to a yes bid, the affect is limited to the bid owner only. As a result, the cost effect of changing a doubtful bid is smaller than the rescheduling function.

d. Quorum

In a real situation, not all invitees are required to attend a meeting. A host and the important invitees should attend, but some invitees are not required to do so. A quorum is used to control the attendance rate of the compulsory and non-compulsory groups. It can make sure that a sufficient number of attendees are at the meeting.

e. Locking mechanism

After a scheduler announces all possible time slots for a meeting, the invitee_agent sends the bid and then locks the available time slots by changing the status from empty to pending. The purpose of locking is to prevent other

meetings from occupying the time slot selected by the scheduler and assigned to the meeting.

2.2.1 Summary of the meeting scheduling

Table 3 summarizes the features of the meeting scheduling systems that are discussed above.

Table 3. Summary of the scheduling features

Feature	Description
Database System Management	1. Decentralized 2. Centralized
Time slot	1. Proposed by host 2. Proposed by invitees
Number of time slots	1. Arbitrary number of time slots 2. Based on a formula 3. All possible time slots within the range given by the host 4. No time slot is used.
Scheduler	1. Decentralized 2. Centralized
Type of bid	1. Yes 2. No 3. Doubtful 4. With other proposed time slots

	5. With the reasons for rejection
Rescheduling a private meeting	Reschedule a private meeting if no common time slot is found
Rescheduling a scheduled meeting	Reschedule a confirmed meeting if no common time slot is found
Quorum	Invitees are separated into groups. The attendance rate of groups should satisfy the minimum requirement.
Locking mechanism	Invitees lock the potential time slots for a meeting. The time slots can be released when the meeting is scheduled to another time slot or it is impossible to assign them for the meeting.

2.3 Mobile Agent Introduction

The term agent means a software program that executes a series of actions on behalf of a user. An agent should perform its assigned work predefined by a developer or a user [10]. An agent can communicate with other agents by conventional means, such as various forms of remote procedure calling and messaging. However, not all agents are mobile agents. If the mobility ability is added to the agent, then the agent becomes mobile and can move from one place to another. Here are the advantages of using mobile agents [30]:

a. Reduce the network load

b. Overcome network latency

c. Encapsulate protocols

d. Execute asynchronously and autonomously

e. Naturally heterogeneous

f. Robust and fault tolerant

CHAPTER 3
ASSIST

In this chapter, we introduce our virtual secretary system (ASSIST). It is a web-based automatic secretary system providing simple secretarial services, such as meeting scheduling, document management, address book and e-mail services. Like other similar virtual secretaries described in chapter 2, our virtual secretary system (ASSIST) focuses on the function of meeting scheduling. Figure 27 shows the system architecture and functions of ASSIST.

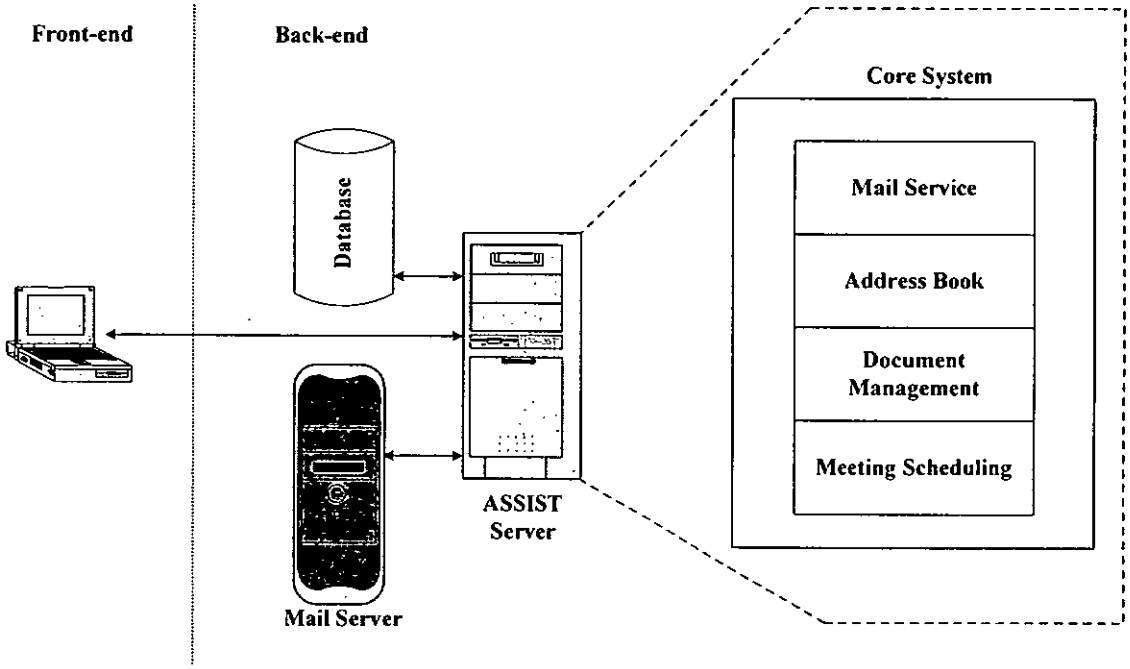

Figure 27. Overview of ASSIST

3.1 System Architecture of ASSIST

As shown in Figure 27, ASSIST is a typical 3-tier system. It can be divided into the front-end system, the business logic server and the back-end system.

- Front-end System

  The front-end system provides a graphical user interface for members to access the system. Users can communicate with ASSIST via a web browser. Requests from the users are transferred to the ASSIST server through the HTTP protocol, and then these requests are handled by the core system. After the core system processes the request, the result is returned to the users through the web server of the ASSIST and displayed as a web page.

- Core System

  The core system processes the request for document management, address book, e-mail service and meeting scheduling. The core system implements the business logic of different functions. In addition, the ASSIST server consists of a web server. The web server is used to provide the essential web services for the users.

- Back-end and Supportive System

  The back-end system is the database for managing the system and storing the necessary data for the core system, such as user profiles, scheduled meeting information etc. The core system accesses the database to get the data for its operations.

3.2 ASSIST's Functions

In this section, we present the possible functions provided by ASSIST. As mentioned before our focus in this thesis is on the meeting scheduling function.

a. Document management

ASSIST's members can use the document management service to access and edit files anywhere under a virtual desktop environment. The table below shows the functions related to the document management.

Table 4 Detailed functions of ASSIST on the document management

Function	Description
File creation	Users can create a file through the ASSIST interface. The file can be stored in the ASSIST server. Later the users can access, edit or download the file anywhere.
File browsing and editing	Users can open them via the interface provided by ASSIST. In addition, users can edit any files under ASSIST server directly through the web interface provided by ASSIST.
File uploading and downloading	This function is similar to the FTP function. In this case, ASSIST acts as an FTP server. Users can upload or download any files to the ASSIST server. Users get or store the files in the ASSIST server without any FTP client, because the required interface is provided by ASSIST. As a result, users can upload or download the

	file through the web browser easily.
Data protection	Only authorized persons can access the data. Furthermore, sensitive, important and confidential data can be encrypted using the document protection function. Only the file owner can view the source data. All unauthorized persons cannot view or download any data from the ASSIST server.
Data backup	Backup is a good practice and necessary action to prevent data lost. ASSIST can duplicate important data and perform periodical backup. In case users delete the source file accidentally, or the local storage system crashes, they can get back the file from the ASSIST server using the latest backup image.
Data compression	For files which are unused for a long period, ASSIST can be used to compress the files automatically in order to release more storage spaces.
Document sharing	Users can share the documents, programs or files under the ASSIST document server. Users allow the authorized person(s) to access the shared documents only.
Data synchronization and version control	In order to provide a convenient working area for the working groups, ASSIST provides the data synchronization and version control function on the documents. Users can keep the source of the documents in the document server. ASSIST keeps the

	latest and few old versions of the documents. Users can get the latest version from the document server or get the history of the document. Also, users can compare the source of the document from the ASSIST server and the version in the local machine. The data synchronization function can be used for updating purposes.

A prototype has been developed to demonstrate some basic functions. In particular ASSIST's users can upload files to the ASSIST document server. Furthermore, users can perform the basic document management functions through the web interface, such as copy or move a file to other directories, delete a directory, compress a file.

b. Web-based address book

ASSIST provides the online address book function. The address book keeps the common information, such as the first name, last name, telephone number, mobile number, email address and home address of the users' friends. Users can access the ASSIST address book through the web browse to get his/her friend's information. As the interface is web-based, users can get the information anywhere and anytime.

c. Electronic mail services

Users can send and check emails using ASSIST. They can also manage different email accounts via a single interface. Also, email filtering and filing

functions can be provided. The table below shows the email functions that may be supported by ASSIST.

Table 5 Detailed functions of ASSIST on the email service

Function	Description
Email checking	Members can check emails from different email accounts via the single web interface of the ASSIST. This function provides a user friendly interface for the members to handle multi-email accounts.
Email sending	Besides checking emails through the ASSIST's interface, users can send emails through the ASSIST web interface. Users can also obtain the recipients' email addresses from the address book kept by the system. Users can send emails by making a phone call or sending a SMS to the ASSIST. To do this, a user provides the receiver's name, email subject and content accordingly. When the system receives the phone call or SMS from the user, it sends the email based on the information provided by the user.
Email filing	ASSIST can file emails into corresponding folders. For example, when a user receives a bill from the mobile phone company or an account statement from a bank, ASSIST can file the email into the bill folder and the statement email into the bank folder.

Elementary email filtering	For filtering emails, members can specify the email addresses, keywords on the email subject or the names of the senders. The filtering service then filters the emails based on the given criteria. As a result, junk emails can be deleted.
Advanced email filtering	Advanced email filtering services can also be provided. In this case, ASSIST identifies the email as a junk mail on behalf of users based on their previous email histories and possibly other members' classifications or actions. ASSIST marks the emails as potential junk mails if the user has marked similar emails as junk mail before based on the sender name, IP address of the mail server or the subject. In addition, the junk mail list of other users can also be consulted. For example, if a large group of users identifies a particular IP address of the mail server as a junk mail source, an email originated from the same source will also be marked "junk". Furthermore, ASSIST can also filter emails if the sender names and email addresses cannot be found in the email address book of the user.
SMS reminder	ASSIST can check the email account(s) of users periodically. If a user receives an important email, e.g. an email from his/her boss, ASSIST notifies the users by using SMS. As the content length of a SMS is

	limited to 160 characters, the SMS contains the sender name and the subject of the email only.

As we know, email is one of the most important and convenient communication tools in this electronic world. We have implemented some of above email functions in the prototype system. Beside the basic checking and sending email functions, we have implemented the email filtering functions as well. In particular, users can use the filtering service to guard the rubbish emails by setting up their own filtering rules based on the sender name, the sender email address and some subject keyword(s). When a user checks emails through the ASSIST email service, ASSIST scans all emails and then filters the emails based on the predefined rules. As a result, when the users read the emails, ASSIST displays only those that are not marked as junk mails.

d. Resource booking and sharing

Meeting rooms, projectors, notebooks and white broads are the common resources when holding a meeting. ASSIST can also provide a resource booking service for users.

e. Meeting scheduling and reminder service

Members can call a meeting and invite other people to attend the meeting. The user provides information on the meeting, such as the topic, preferred meeting dates, invitees' names, etc. The meeting scheduling system then proposes some time slots for the meeting and informs all invitees to choose the time slots for which they are available. Finally, the system can choose the most

suitable time slot for the meeting while meeting a predefined criterion, e.g., the overall attendance rate. Later we will discuss the meeting scheduling protocol in details in chapter 4 and 5. Finally, a reminder service can be used to alert the host and all attendees to attend the meeting by telephone (e.g. 30 minutes before it is scheduled to start).

The meeting scheduling service is the most important function in ASSIST. We have implemented the complete meeting scheduling process. As shown in Figure 28, the host of the meeting initiates a new meeting through the ASSIST meeting scheduling function in step 1. The host should provide the basic information of the meeting, such as the meeting title, duration and invitees. Having received the information, ASSIST uses the meeting scheduling algorithm (which will be discussed in chapter 4 and 5) to find the optimal number of time slots (see step 2). After the calculation process, ASSIST selects the proposed time slots and then sends the information to the host and all invitees by email (see step 3). The host and the invitees should logon to the ASSIST system and then reply to the system whether they are available to attend the meeting at the specified time slots. User can specify the preference on each time slot, as shown in step 4. After the host and all invitees reply to the system with the preferences on the time slots, ASSIST chooses the best and most suitable time slot for the meeting (see step 5). The confirmed time slot is sent to the host and invitees by email in step 6. The confirmed time slot is updated in the personal organizer of the users. In addition, reminder service provide by ASSIST can remind the invitees to attend the meeting, such as 30 minutes before the meeting starts by an automatic phone call (see step 7).

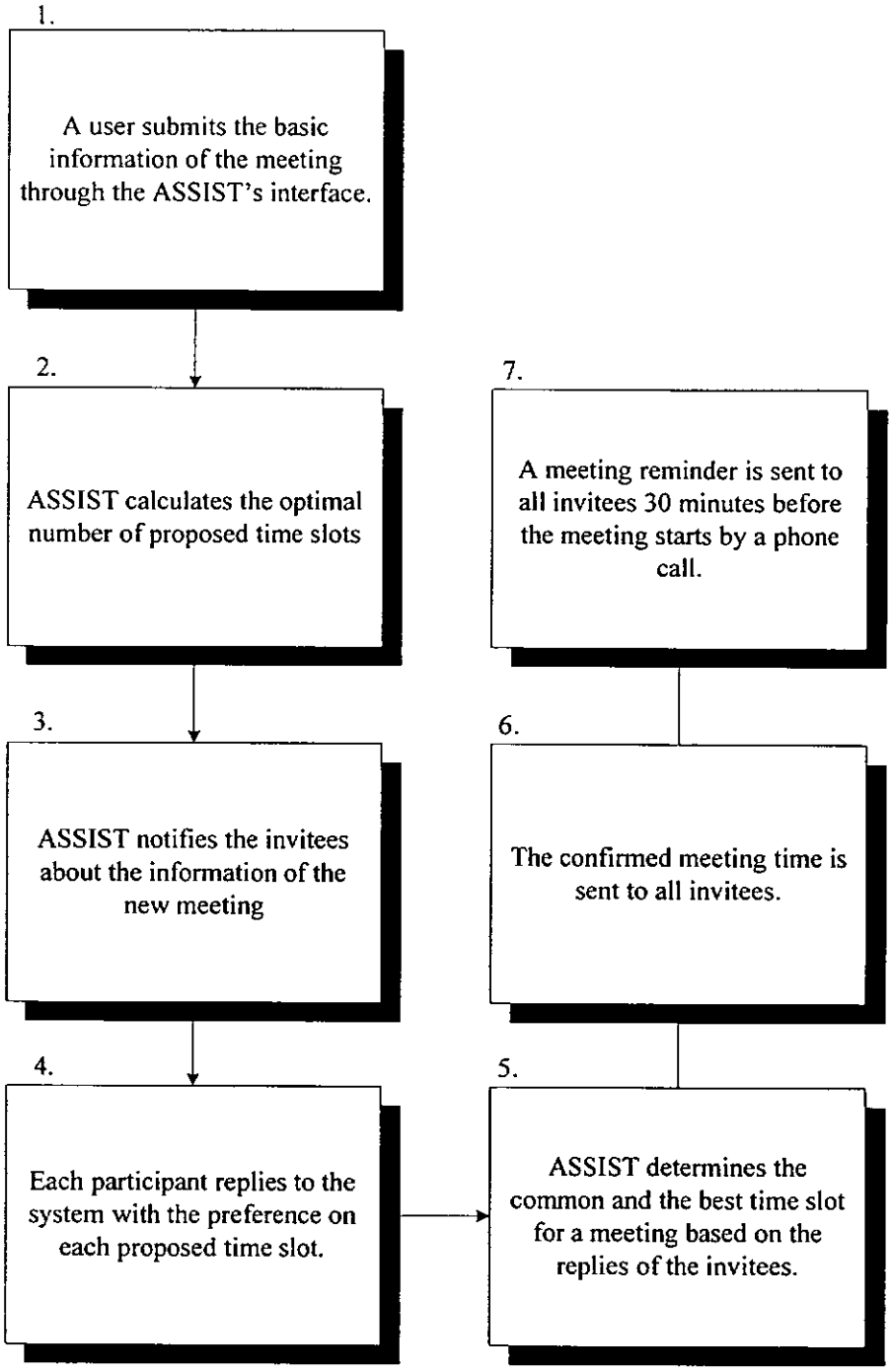

Figure 28. Logic flow of the meeting scheduling function

3.3 Communication Between ASSISTs

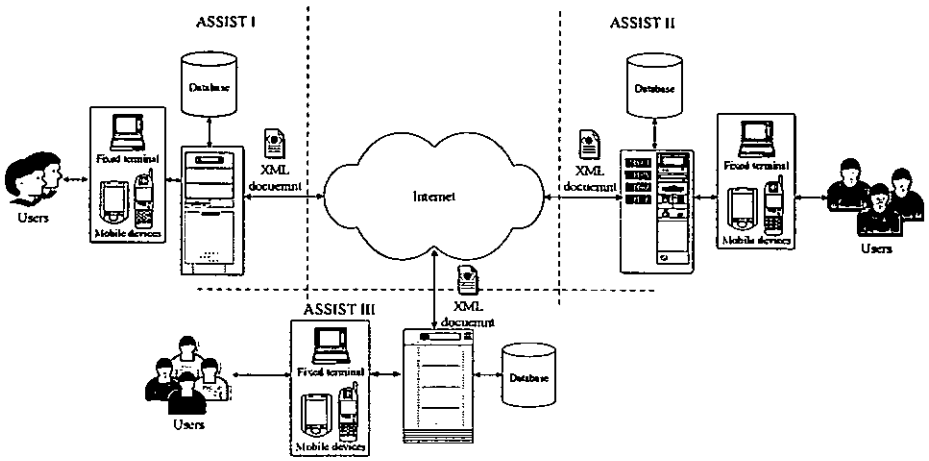

Figure 29. Overview of the system architecture

Figure 29 shows the system architecture of the virtual secretaries. Users can access the system through a web browser or any mobile device, such as a PDA and a WAP phone. Users can initiate a new meeting or check existing scheduled meetings through the virtual secretaries. An automatic secretary system can communicate with other automatic secretary systems by exchanging XML-based messages (e.g., for scheduling meetings).

Figure 30 shows the general meeting scheduling mechanism within a single meeting scheduling system. After receiving a request, the system calculates the number of proposed time slots according to the information provided by the host. The server then sends the time slots to the invitees and locks in those time slots. Furthermore, the server waits for the invitees to reply on each proposed time slot (possibly including their preferences). Having received all of the responses, the system determines the

appropriate time slot for the meeting and informs the host and invitees accordingly. Finally, the unused time slots are released.

As shown in Figure 30, the meeting scheduling system communicates with the host and invitees by means of XML documents. The advantage of using XML document is to provide a standard communication format for exchanging information between the server and users, as well as among different meeting scheduling systems. Figure 31 shows how the automatic secretaries exchange information.

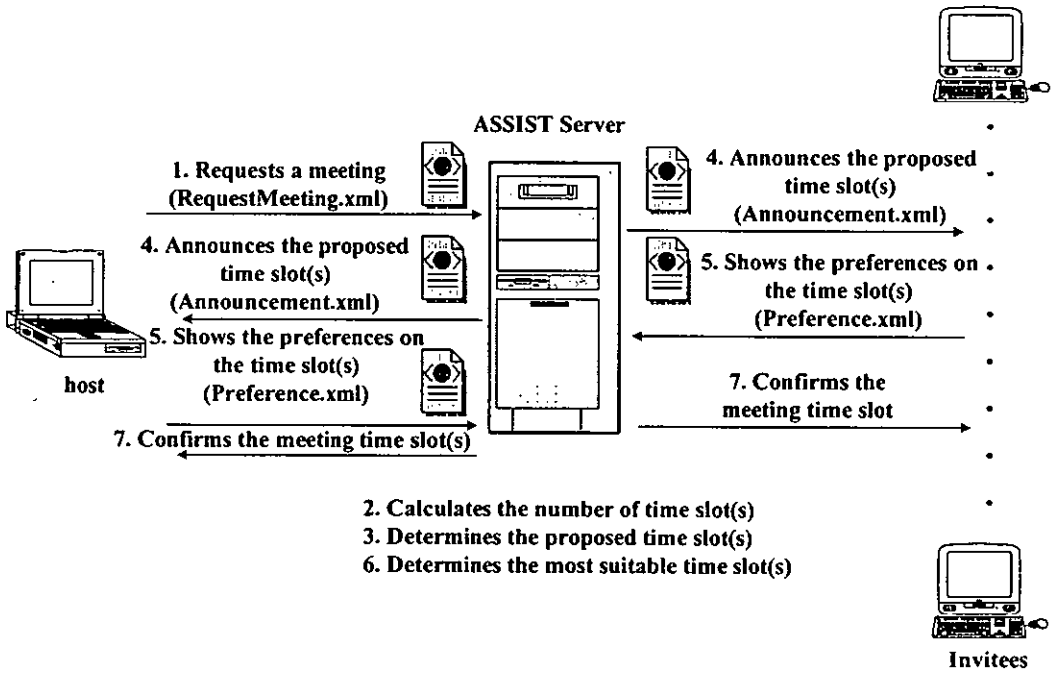

Figure 30. Meeting scheduling workflow of a standalone ASSIST

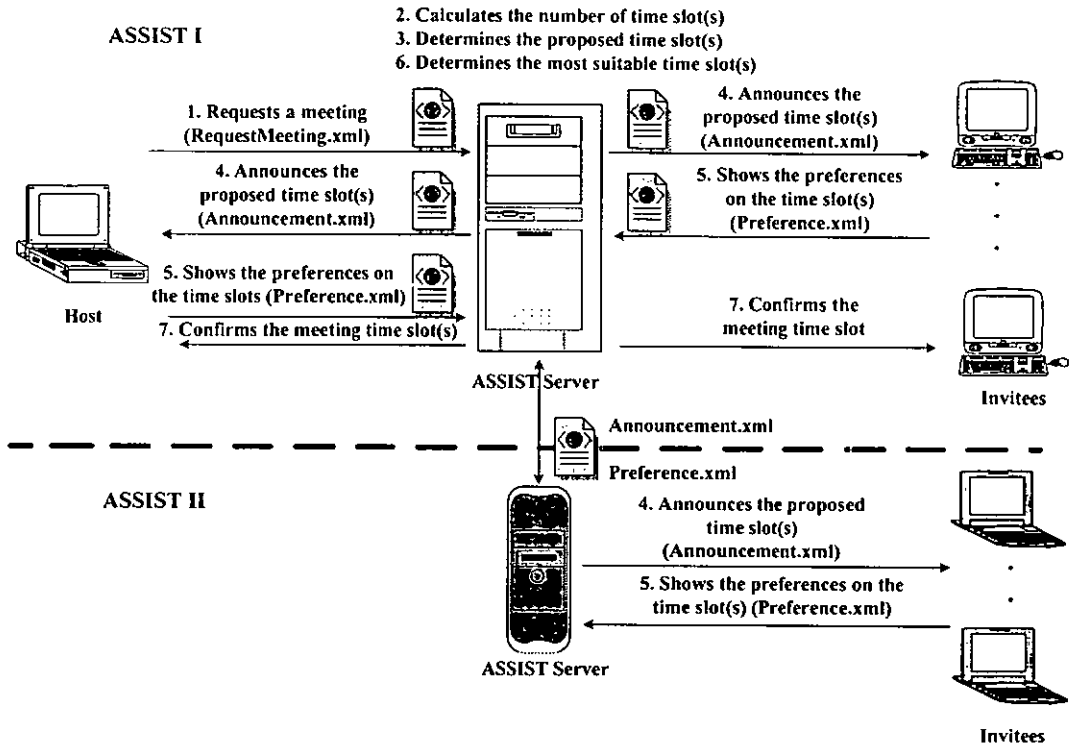

Figure 31. Communication between two ASSISTs

XML examples for communicating between users

1. Request a meeting using RequestMeeting.xml

The XML document for requesting a new meeting contains all the information about a meeting, including the topic of the meeting, duration of the meeting, preferred meeting date, name of invitees etc. After generating the XML document for a request, the ASSIST server parses the meeting request XML document and extracts the meeting information from the XML document. The DTD for the meeting-request XML document is shown in Figure 32.

```
<!ELEMENT RequestMeeting
(MeetingID,Member,Group,Topic,Duration,Date,Description,
Probability,Days,Period)+>
<!ELEMENT MeetingID (#PCDATA)>
<!ELEMENT Member (#PCDATA)>
<!ELEMENT Group (#PCDATA)>
<!ELEMENT Topic (#PCDATA)>
<!ELEMENT Duration (#PCDATA)>
<!ELEMENT Date (From,To)?>
<!ELEMENT From (#PCDATA)>
<!ELEMENT To (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT Probability (#PCDATA)>
<!ELEMENT Days (#PCDATA)>
<!ELEMENT Period (fromH,fromMin,toH,toMin)?>
<!ELEMENT fromH (#PCDATA)>
<!ELEMENT fromMin (#PCDATA)>
<!ELEMENT toH (#PCDATA)>
<!ELEMENT toMin (#PCDATA)>
```

Figure 32. DTD of RequeestMeeting.xml

As shown in Figure 32, the meeting id, name of the invitees, group name, topic, duration, meeting date, description, probability that schedule a meeting successfully, day of week, and the suggested time range for the meeting form the elements of the request XML document. Figure 33 shows the corresponding XML document.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE RequestMeeting SYSTEM "RequestMeeting.dtd">
<RequestMeeting>
        <MeetingID>57</MeetingID>
        <Member>ivan,irene</Member>
        <Group></Group>
        <Topic>AGM1</Topic>
        <Duration>1</Duration>
        <Date>
                <From>12-15-2002</From>
                <To>12-23-2002</To>
        </Date>
        <Description>N/A</Description>
        <Probability>0.95</Probability>
        <Days>2,3,4,5,6,7</Days>
        <Period>
                <fromH>9</fromH>
                <fromMin>0</fromMin>
                <toH>19</toH>
                <toMin>0</toMin>
        </Period>
</RequestMeeting>
```

Figure 33. Example of RequestMeeting.xml

2. Inform the proposed time slot to the host and invitees using Announcement.xml

```
<!ELEMENT Announcement
(meetingid,member,topic,description,duration,timeslot+)>
<!ELEMENT meetingid (#PCDATA)>
<!ELEMENT member (#PCDATA)>
  <!ELEMENT topic (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT duration (#PCDATA)>
<!ELEMENT timeslot (starttime,endtime)?>
<!ATTLIST timeslot id CDATA #REQUIRED>
<!ELEMENT starttime (#PCDATA)>
<!ELEMENT endtime (#PCDATA)>
```

Figure 34. DTD of Announcement.xml

The ASSIST server parses the meeting-request XML document after the meeting host sends a request to the server. Afterwards, the ASSIST server generates a new XML document (Announcement.xml) with the meeting information for asking the meeting invitees to attend the meeting. Then the ASSIST server sends the XML document to the meeting host and invitees. Figure 34 shows the DTD of the Announcement.xml, Table 6 shows the meaning of the elements and attributes in the DTD of the Announcement.xml document and Figure 35 shows the example of the Announcement.xml.

Table 6. Elements and attributes in Announcement.xml

Element/Attribute name	**Meaning**
meetingid	The meeting ID for a requested meeting
member	User ID of a meeting who are invited
topic	Topic of the meeting
description	Any information of the meeting
duration	Duration of the meeting
timeslot	Start time and the end time of the proposed time slot and id is the attribute of the timeslot
id	ID of the time sots
starttime	The start time of the proposed time slot
endtime	The end time of the proposed time slot

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Announcement SYSTEM "Announcement.dtd">
<Announcement>
        <member>derek</member>
        <meetingID>109</meetingID>
        <topic>Annual General Meeting</topic>
        <description>N/A</description>
        <duration>1</duration>
        <timeslot id="192">
                <starttime>30/12/2002 09:00:00</starttime>
                <endtime>30/12/2002 10:00:00</endtime>
        </timeslot>
        <timeslot id="193">
                <starttime>30/12/2002 10:00:00</starttime>
                <endtime>30/12/2002 11:00:00</endtime>
        </timeslot>
        <timeslot id="194">
                <starttime>30/12/2002 11:00:00</starttime>
                <endtime>30/12/2002 12:00:00</endtime>
        </timeslot>
</ Announcement >
```

Figure 35. Example of Announcement.xml

3. Fill in the preference on each proposed time slot using Preference.xml

The host and invitees receive and parse the Announcement.xml. They get the meeting topic, description, duration of meeting and the proposed time slots from the XML document. Afterwards, the host and invitees check their schedules and give the score on each proposed time slot. The score is used to indicate the preference of the invitee to attend the meeting at the particular time slot. The invitee gives a high score if he/she prefers to attend the meeting at the time slot. Finally, the ASSIST server based on the scores generates the XML document called Preference.xml. Figures 36 and 37 show the DTD of the Preference.xml and the example of the Preference.xml respectively. Table 7 shows the meaning of the elements and attributes in the DTD of the Preference.xml.

When the ASSIST server receives the scores of the proposed time slot from the host and the invitees, it parses and validates the replied XML document. The server then gets the score of the time slots from the XML document and stores the data into the database. After the ASSIST server receives all scoring replies from the host and invitees, it determines the common time slot with the highest score for the meeting.

```
<!ELEMENT RespondMeeting (userid,meetingid,selectedtimeslots+)>

<!ELEMENT userid (#PCDATA)>

<!ELEMENT meetingid (#PCDATA)>

<!ELEMENT selectedtimeslots (score)?>

<!ATTLIST selectedtimeslots value CDATA #REQUIRED>

<!ELEMENT score (#PCDATA)>
```

Figure 36. DTD for Preference.xml

Element/Attribute name	Meaning
meetingid	The meeting ID for a requested meeting
selectedtimeslots	Time slot proposed for the meeting, it has the attribute value
value	The ID of the proposed time slots
score	The score of the proposed time slot, it can reflect the acceptability of the time slot
duration	Duration of the meeting

Table 7. Elements and attributes used in Preference.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Preference SYSTEM "Preference.dtd">
<Preference>
        <meetingid>109</meetingid>
        <selectedtimeslots value="192">
                <score>1</score>
        </selectedtimeslots>
        <selectedtimeslots value="193">
                <score>1</score>
        </selectedtimeslots>
        <selectedtimeslots value="194">
                <score>3</score>
        </selectedtimeslots>
</Preference>
```

Figure 37. Example of Preference.xml

3.4 Summary

In this chapter, an automatic secretary system called ASSIST is proposed for handling secretarial tasks in general and scheduling meetings in particular. The architecture and system functions are introduced. Essentially ASSIST provides document management, email service, address book handling, meeting scheduling and reminder service. A prototype is developed to demonstrate some of these functions.

CHAPTER 4
MEETING SCHEDULING

Most of the existing meeting scheduling mechanisms just assign an arbitrary number of time slots for scheduling purposes. In this chapter, we determine the optimal number of time slots based on a mathematical model.

4.1 Scheduling Workflow

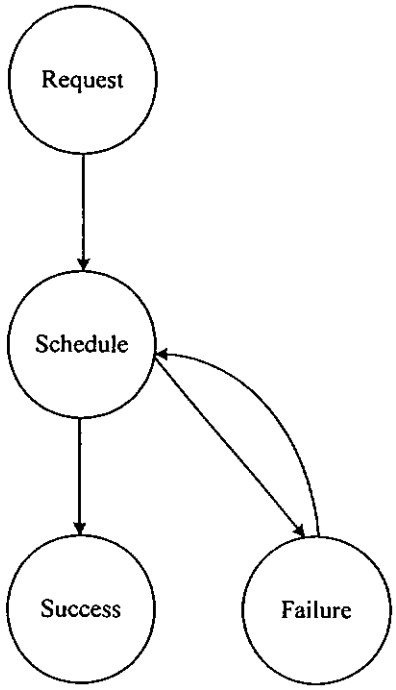

Figure 38. Stage diagram for meeting scheduling

As shown in Figure 38, when a user requests a meeting, the system starts to schedule a time slot for the meeting (in the scheduling stage). The system proposes certain time slots to the invitees. If a common available time slot is found within those time slots,

then the meeting is scheduled at that time slot. The stage is then changed from "schedule" to "success." However, if no common available time slot can be found among those time slots, the scheduling process is unsuccessful. The system will continue the scheduling process by suggesting another set of time slots to the invitees. The above process repeats until a free time slot is found. Finally, the state of success should be reached.

The scheduling cost depends on the number of time slots used in each iteration and on the total number of iterations. The scheduling cost will be determined by the total amount of time needed to schedule a meeting, usage of the system's resources and the total amount of time that invitees spend giving their replies.

Examples

Let us look at a simple example. A is the host of the meeting and he/she invitees B, C and D to attend the meeting. We assume A, B, C and D are in the same group and that all of them should participate in the meeting. The meeting lasts for an hour. Figure 39 shows the schedules of the host and the invitees.

9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00
A	×			Lunch			×				
B		×	×	Lunch							
C				Lunch	×		×				
D				Lunch		×					

Legend: × : occupied time slot; ▨ : proposed time slot; ✱ : scheduled time slot

Figure 39. Schedules of A, B, C and D

Scenario I: Few time slots and more iterations

The system provides a few time slots in each iteration for the invitees. However, if no common time slot can be found in an iteration, the system needs to start a new iteration by providing another set of time slots. The iteration continues until the meeting can be scheduled successfully.

1. The first iteration

We assume that the system provides two time slots in each round. Figure 40 shows the time slots proposed by the system (as highlighted) in the first iteration.

	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00
A		×			Lunch			×				
B			×	×	Lunch							
C					Lunch	×		×				
D					Lunch		×					

Legend	× : occupied time slot	: proposed time slot	✸ : scheduled time slot

Figure 40. Two time slots from 10:00 to 12:00 selected by the scheduling system

Since B is not available for the proposed time slots (from 10:00 to 12:00), the meeting cannot be scheduled. As a result, the system needs to start the second iteration by proposing another two time slots.

Note that the system does not propose the time slot from 09:00 to 10:00, because host A is not available at that time slot. Basically, the system proposes the earliest available time slot(s) of the initiator.

2. The second iteration

In the second iteration, the system chooses another two time slots. The system selects the time slots from 13:00 to 14:00 and 14:00 to 15:00. The scheduling system skips the time slot from 12:00 to 13:00, because it is the lunch hour for everyone.

9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00
A	×			Lunch		×				
B		×	×	Lunch						
C				Lunch	×	×				
D				Lunch		×				

Legend	× : occupied time slot	: proposed time slot	✸ : scheduled time slot

Figure 41. Selected time slots in the second iteration of the meeting scheduling

However, the meeting cannot be scheduled because C is not available at the time slot from 13:00 to 14:00, and D is not available from 14:00 to 15:00. As a result, the scheduling system needs to start the third iteration.

3. The third iteration

The scheduling system provides another two time slots to the invitees in the third iteration. As shown in Figure 42, the time slots from 16:00 to 17:00 and 17:00 to 18:00 are selected as the proposed time slots.

9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00
A	×			Lunch			×	✸		
B		×	×	Lunch				✸		
C				Lunch	×		×	✸		
D				Lunch		×		✸		

Legend	× : occupied time slot	: proposed time slot	✸ : scheduled time slot

Figure 42. The two time slots selected in the third iteration of the scheduling

The meeting is scheduled for the time slot from 16:00 to 17:00, because both the host of the meeting and the invitees are free at that particular time slot.

In the above example, three iterations were needed, each with two time slots, to schedule the meeting successfully.

Scenario II: More time slots and few iterations

In the second scenario, the scheduling system provides more time slots in each iteration.

1. The first iteration

The system provides eight time slots in the first iteration as shown below in Figure 43.

9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00
A	×			Lunch			×	✸			
B		×	×	Lunch				✸			
C				Lunch	×		×	✸			
D				Lunch		×		✸			

Legend	× : occupied time slot	: proposed time slot	✸ : scheduled time slot

Figure 43. Eight time slots selected in the first iteration

The system selects eight time slots from 10:00 to 12:00, 13:00 to 15:00 and 16:00 to 20:00. It skips the time slots from 12:00 to 13:00 and 15:00 to 16:00, because the meeting's host is not available at those time slots. Finally, the meeting is scheduled at the time slot from 16:00 to 17:00 because this is the earliest common free time slot.

Scenarios I and II show the two approaches to scheduling a meeting. The first scenario uses fewer time slots in each iteration, but more iterations are needed, while the latter scenario uses more time slots and only one iteration.

When the number of iterations increases, the total time spent on scheduling a meeting will also increase. Furthermore, if the number of time slots proposed in each iteration increases, the time spent on checking the schedules and finding a free common time slot will increase. In addition, the system needs to lock in the proposed time slots (i.e., so that they will not be available for scheduling other meetings). Hence, both the number of iterations and the number of time slots per iteration affect the cost of scheduling.

Obviously, there is a tradeoff between scenarios I and II. If the system provides more time slots in each iteration, the meeting can be scheduled easily and fewer iterations would be required. However each invitee needs to fill in more time slots per iteration. If fewer time slots are used in each iteration, more iterations may be required. In the next section, a mathematical model is proposed to find the best number of time slots in order to minimize the cost of scheduling.

4.2 Mathematical Model

The aim of the mathematical model is to find the optimal number of time slots when scheduling a meeting. We know from the discussion in Section 4.1 that there is a tradeoff between the number of proposed time slots in each iteration and the total number of iterations when scheduling a meeting. As a result, we formulate the

meeting scheduling problem using the mathematical model, in order to obtain a solution that optimally balances the number of proposed time slots and the number of iterations.

4.2.1 Average Cost of Scheduling

In this section, a model is set up to find the optimal number of time slots by calculating the cost of scheduling. As a meeting may last for a few sessions, the system should locate a time slot that includes all sessions of the meeting and which allows all the invitees to attend the whole meeting. The invitees are assigned into different groups. In order to ensure enough invitees attend the meeting, each group should have a minimum attendance rate requirement. That means a minimum number of invitees should attend the meeting. The number of minimum attendees is affected by the minimum attendance rate requirement and the number of invitees in a group. If an insufficient number of attendees in a group are willing to attend a meeting in a particular time slot, that meeting cannot be assigned to that time slot. When the system proposes the time slot(s) to the host and invitees, all should respond by checking their schedules and filling in the time slot(s) to indicate their availability or unavailability for the proposed time(s). When the host or invitees specify their availability by filling in a time slot, the action causes a fill-in cost. If the system cannot find a common time slot after proposing a set of time slots, the system should propose another set of time slots. Once again, host and invitees should check their schedules to determine whether they can attend the meeting at the proposed time(s). The re-proposal of a time slot causes overhead costs in that hosts and invitees will all begin to tire of the system repeatedly enquiring of their availabilities. Based on the fill-in cost on the proposed time slot(s) and the overhead cost caused by each

scheduling cycle, the mathematical model can determine the scheduling cost. Furthermore, the mathematical model also considers the availability of the invitees. The system determines the availability of the invitees based on the schedules of the invitees in the database or on the meeting scheduling history. By minimizing scheduling costs, the model can find the optimal number of time slots when scheduling a meeting [23]. A detailed explanation of the model is provided in this chapter of the thesis.

Table 8. Notations used in the model

Symbol	Meaning
m	The number of the time slots proposed in each iteration
i	The i^{th} iteration
Q	Probability that a meeting can be scheduled at the i^{th} iteration
f	Cost of filling in a time slot
o	Overhead cost of each iteration of each invitee
p	Probability that an invitee is free at a time slot
d	Required number of sessions for a meeting
q	Probability that an invitee can attend the meeting
G	Number of groups
j	The j^{th} meeting group
A_j	Minimum attendance rate of group M_j
I_j	Number of people in group M_j
N_j	Minimum number of attendees in group M_j

For simplicity, we assume the following:

- The required iterations can be completed before the scheduling deadline.
- Q, o and f are constant and are not affected by the number of iterations.
- The probability that an invitee is free at a time slot for all invitees.
- Each person can only belong to one group.

When scheduling a meeting (at the first iteration), m time slots are proposed by the system. The meeting can be scheduled successfully with probability Q. In other words, none of the m time slots are suitable for the meeting with probability $(1 - Q)$. As a result, the scheduling process enters into the second iteration from the first iteration with probability $(1 - Q)$ as shown in Figure 44.

Figure 45 shows that when a user initiates a meeting, the system proposes m time slots, locks in those time slots and sends the m choices to the invitees. When invitees receive the request, they (or their automatic secretaries) check the schedules and replies to the systems with the available time slots.

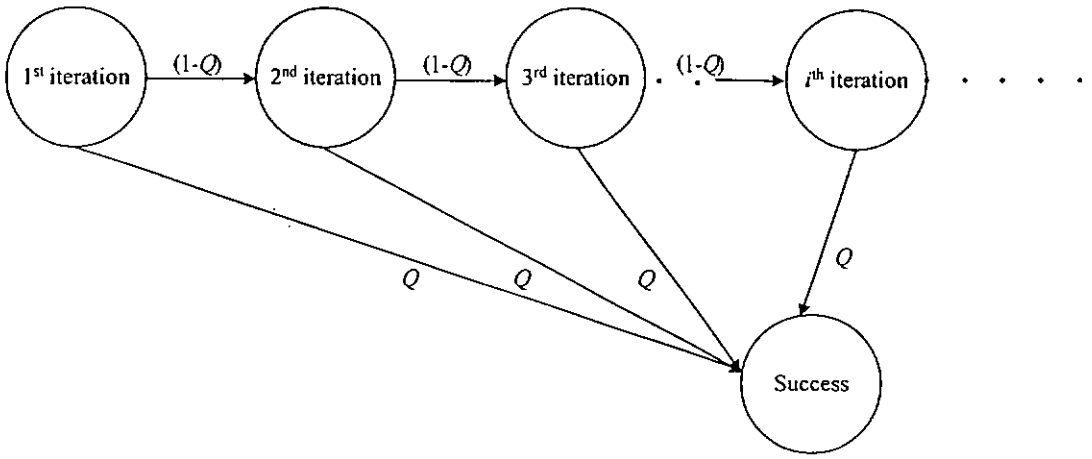

Figure 44. Stage diagram to show that the meeting scheduling process

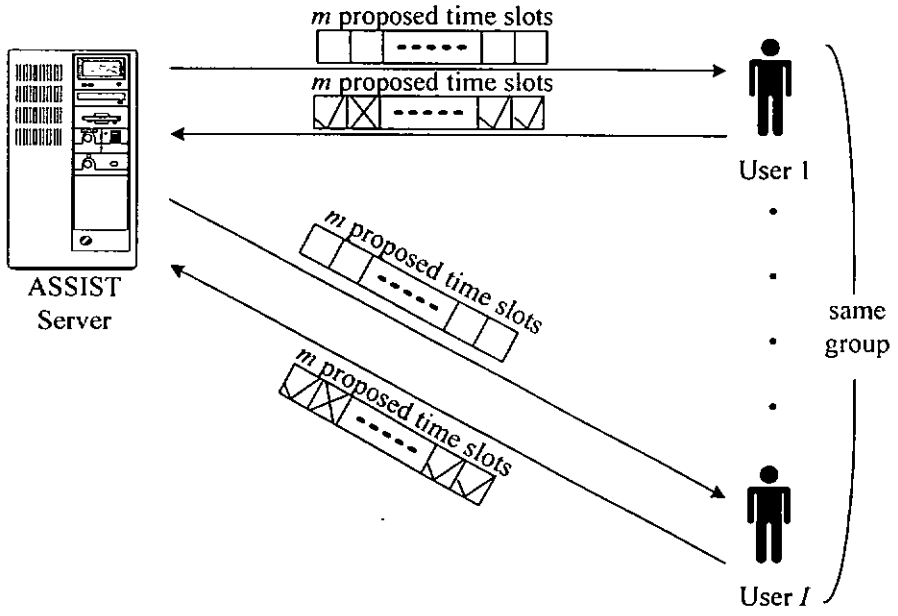

Figure 45. The server sends the proposed time slots to the users and the users give their replies

Table 9. m, Q and cost in each iteration

No. of iterations	Number of time slots	Probability that the meeting can be scheduled successfully at the i^{th} iteration	Cost of scheduling a meeting successfully
1	M	Q	$\sum_{j=1}^{G} I_j(mf+o)$
2	M	$(1-Q)Q$	$2\sum_{j=1}^{G} I_j(mf+o)$
3	M	$(1-Q)^2Q$	$3\sum_{j=1}^{G} I_j(mf+o)$
i	M	$(1-Q)^{i-1}Q$	$i\sum_{j=1}^{G} I_j(mf+o)$

In the first iteration, the cost of filling in the m proposed time slots is mf and the overhead cost for each invitee is o. Therefore, the total cost of the first iteration is $mf+o$. If the meeting cannot be scheduled in the first iteration, the system will start the

second iteration. The accumulated cost in the second iteration is $2(mf + o)$. Table 9 lists the accumulated cost of each iteration and the probability of scheduling a meeting successfully at a particular iteration.

According to Table 9, the average cost of scheduling a meeting is shown in equation (3).

Average cost

$$= Q\sum_{j=1}^{G} I_j(mf + o) + 2(1-Q)Q\sum_{j=1}^{G} I_j(mf + o) + \ldots\ldots + i(1-Q)^{i-1}Q\sum_{j=1}^{G} I_j(mf + o)$$

$$= \sum_{k=1}^{i}\left[k(1-Q)^{k-1}Q\sum_{j=1}^{G} I_j(mf + o)\right] \quad (3)$$

The meeting scheduling process continues until a free time slot can be found. Based on equation (3), we have

$$\text{Average cost} = \sum_{k=1}^{\infty}\left[kQ(1-Q)^{k-1}\sum_{j=1}^{G} I_j(mf + o)\right] \quad (4)$$

Denotes the overhead cost of each iteration as $C_{itl} = \sum_{j=1}^{G} I_j(mf + o)$. Based on equation (4), we obtain

$$= \sum_{k=1}^{\infty}\left[k(1-Q)^{k-1}Q\sum_{j=1}^{G} I_j(mf + o)\right]$$

$$= \sum_{k=1}^{\infty}\left[k(1-Q)^{k-1}QC_{itl}\right]$$

$$= \frac{C_{ttl}}{Q} \tag{5}$$

By substituting $C_{ttl} = \sum_{j=1}^{G} I_j(mf + o)$ into equation (5), the average cost of scheduling a meeting is

$$\text{Average cost} = \frac{\sum_{j=1}^{G} I_j(mf + o)}{Q} \tag{6}$$

4.2.2 *Probability Q (probability that the meeting will be scheduled successfully)*

Denote Q as the probability that the meeting can be scheduled. In other words, a free time slot cannot be found for the meeting with probability (1-Q). Q is related to the availability of the invitees, the number of groups and the rate of attendance. We evaluate Q based on the above model.

Recall that p is the probability that an invitee is free at a particular time slot. If the meeting lasts for d sessions, i.e d time slots, the probability that an invitee is free to attend the whole meeting will be

$$q = p^d \tag{7}$$

If the meeting involves G group, the minimum attendance rate of group G_j is A_j and the number of invitees in group G_j is I_j, the minimum number of attendees for group G_j is

$$N_j = \left\lceil I_j \times A_j \right\rceil \tag{8}$$

As a result, the probability that the group will satisfy the minimum attendance rate is

$$\sum_{k=N_j}^{I_j} \binom{I_j}{k} q^k (1-q)^{I_j-k} \tag{9}$$

To schedule the meeting, all G groups must satisfy the attendance requirements. The respective probability P(Success) is

$$\text{P(Success)} = \prod_{j=1}^{G} \left[\sum_{k=N_j}^{I_j} \binom{I_j}{k} q^k (1-q)^{I_j-k} \right] \tag{10}$$

In other words, the probability that a time slot cannot be found is

P(Failure) = 1 – P(Success), which is equal to

$$1 - \prod_{j=1}^{G} \left[\sum_{k=N_j}^{I_j} \binom{I_j}{k} q^k (1-q)^{I_j-k} \right] \tag{11}$$

As mentioned in the previous section, m time slots are proposed by the system to the invitees in each iteration. Therefore, the probability that a meeting cannot be scheduled to one of the m time slots is

$$\left[1 - \prod_{j=1}^{G} \left[\sum_{k=N_j}^{I_j} \binom{I_j}{k} q^k (1-q)^{I_j-k} \right] \right]^m \tag{12}$$

Recall that Q is the probability that a meeting can be successfully scheduled at one of the m proposed slots. Therefore, Q is equal to equation (13).

$$Q = 1-\left[1-\prod_{j=1}^{G}\left[\sum_{k=N_j}^{I_j}\binom{I_j}{k}q^k(1-q)^{I_j-k}\right]\right]^m \tag{13}$$

After obtaining the value of Q, the average cost of scheduling a meeting can be found as follows.

$$\text{Average cost} = \frac{\sum_{j=1}^{G} I_j(mf+o)}{1-\left[1-\prod_{j=1}^{G}\left[\sum_{k=N_j}^{I_j}\binom{I_j}{k}q^k(1-q)^{I_j-k}\right]\right]^m} \tag{14}$$

4.3 Results and Findings

Based on the above meeting scheduling model, we can calculate the average cost according to (14). Furthermore, we can determine the optimal number of time slots. Table 10 shows the base values of the parameters used in the following analysis.

Table 10. Value of parameters used in the model

Parameter	Meaning	Value
G	Number of groups	1
I_1	Number of invitees in group 1	10
f	Cost of filling in a time slot	1
o	Overhead cost of each iteration	10

p	Availability (The probability that an invitee is free at a time slot)	0.8
d	Duration of the meeting in terms of number of slots	1
A_1	Required attendance rate of group 1	75%

Figure 46 shows that the optimal number of time slots increases dramatically when the availability decreases from 0.5 to 0.3. By contrast, the optimal number of time slots remains relatively constant when the availability increases from 0.6 to 1.0.

Figure 46 also shows the number of iterations needed to schedule a meeting. When the availability is equal to 0.3, seven iterations are required. On the other hand, only two iterations are required if the availability is between 0.5 and 0.9. The average number of iterations increases sharply when the availability is less than 0.4.

It can be seen that if the system provides four time slots for the invitees in each iteration, the system can schedule a meeting within two iterations if the availabilities of the invitees are greater than 0.6.

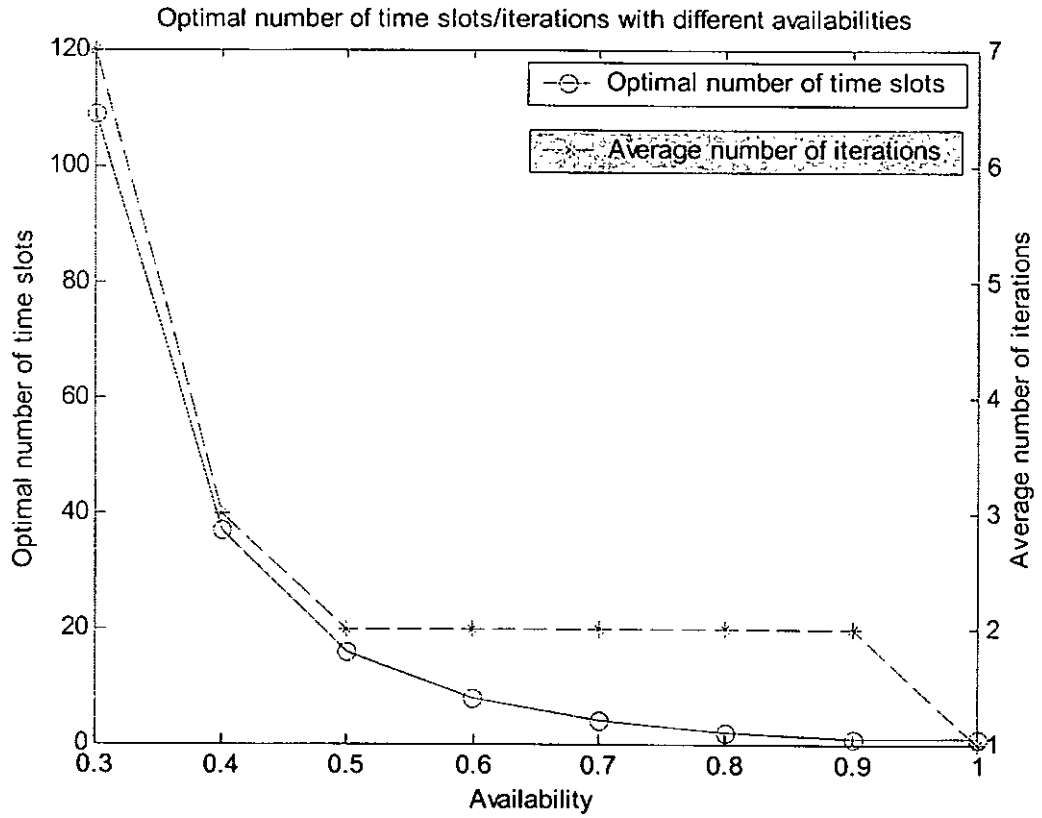

Figure 46. Optimal number of time slots/Average number of iterations with different availabilities

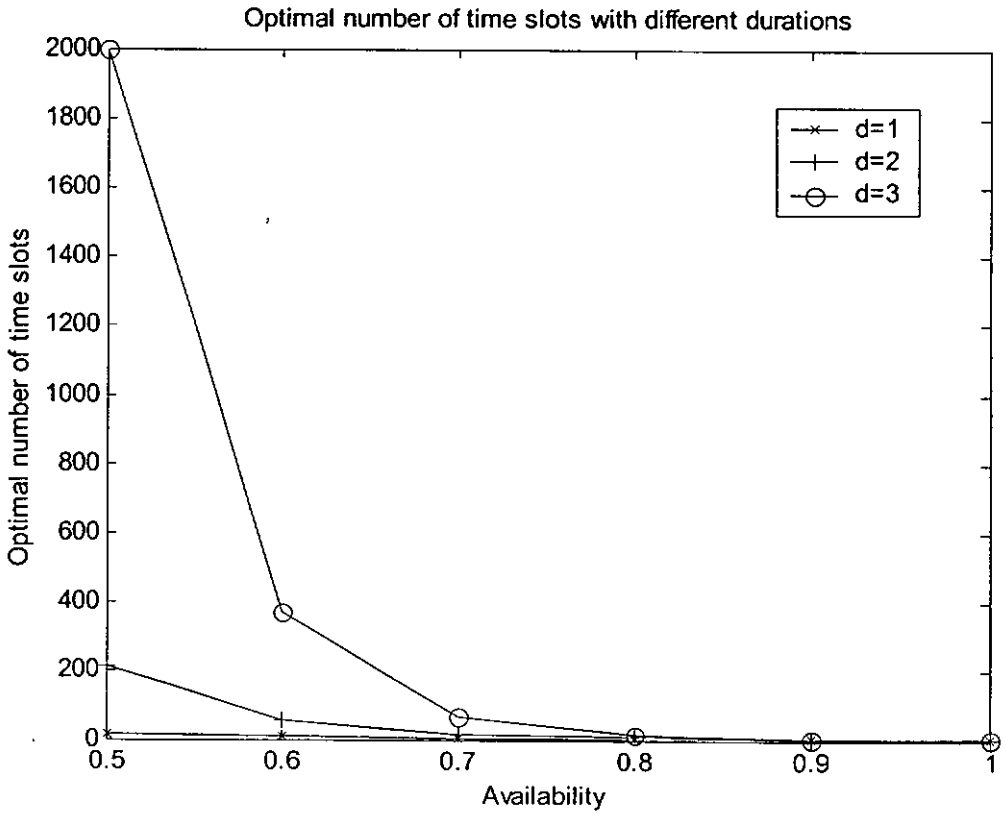

Figure 47. Optimal number of time slots with different meeting durations

Figure 47 shows how the duration of a meeting affects the optimal number of time slots per iteration. Obviously, when the duration of the meeting increases, the optimal number of time slots per iteration should also increase. As shown in Figure 47, the optimal number of time slots increases slightly when the availability is between 0.8 and 1.0 and the duration of the meeting is between one to three time slot(s). However, when the availability is less than 0.8, the optimal number of time slots for scheduling a meeting with a duration of three time slots increases dramatically, especially when the availability is greater than 0.7.

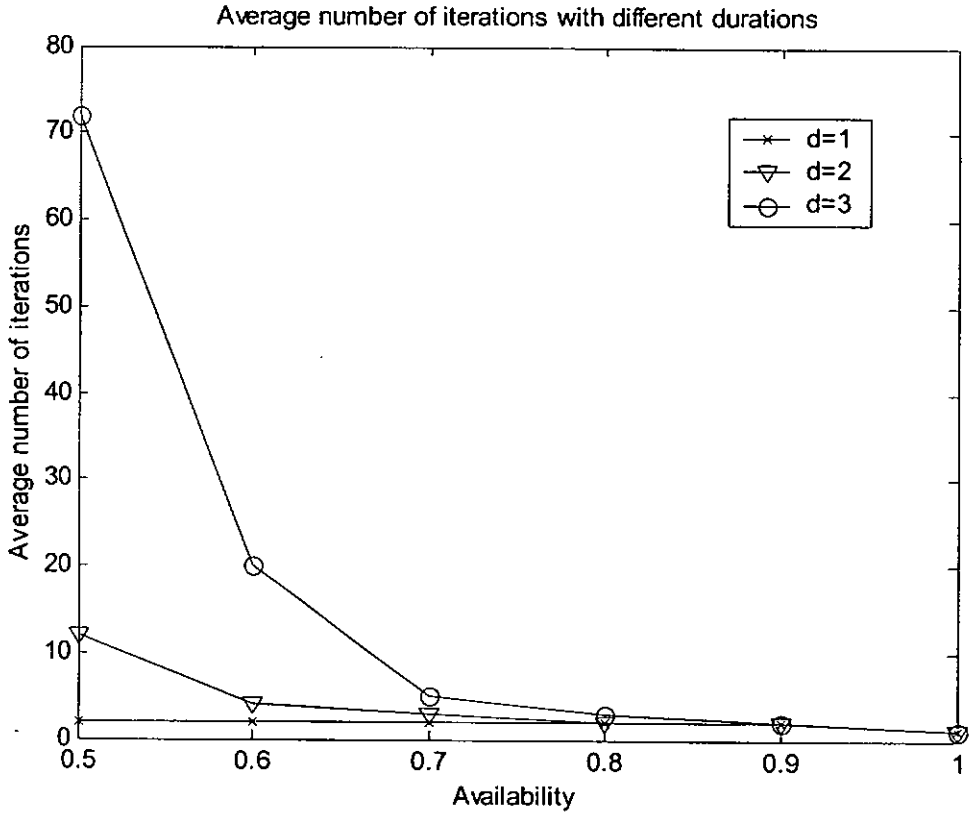

Figure 48. Average number of iterations with different meeting durations

Figure 48 shows that the average number of iterations increases sharply when the availability decreases from 0.7 to 0.5 and the duration of the meeting is equal to three time slots. However, the number of iterations remains at two to three when the availability is between 0.7 and 1.0 and the meeting duration is one and two time

slot(s). When the duration of the meeting increases and the availability decreases at the same time, the cost of scheduling will increase greatly. As a result, the optimal number of time slots and number of iterations will increase. It can be seen from the figure that when the duration of a meeting is between one and three time slots and the availability is greater than 0.7, the cost of scheduling the meeting can be kept at low. In other words, the number of time slots and iterations required are small. However if the duration of the meeting is increased to three time slots and the availability is below 0.7, the cost of scheduling the meeting increases dramatically. This is because, in this case, the availability becomes p^3, which is very small. As a result, the system needs to propose many time slots in each iteration to satisfy the required attendance rate.

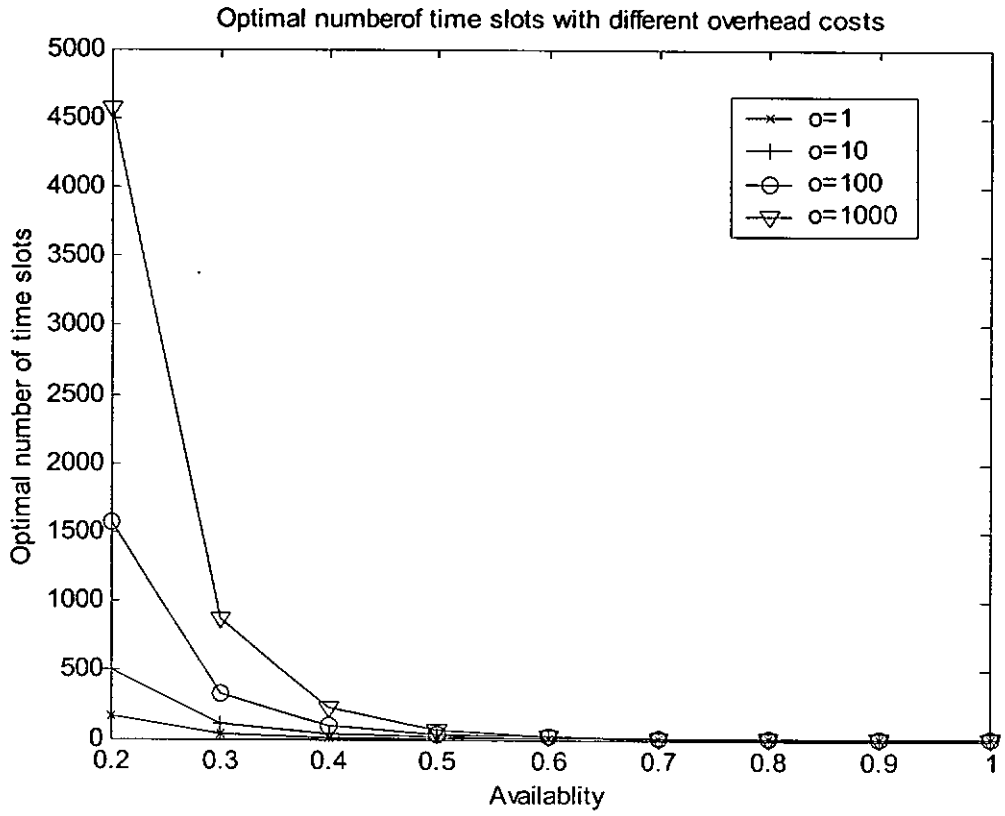

Figure 49. The optimal number of time slots with different overhead costs

We observe that the optimal number of time slots increases when the overhead cost of an iteration increases. As shown in Figure 49, although the overhead cost increases, the optimal number of time slots increases slightly when the availability is greater than 0.6. However, the optimal number of time slots increases dramatically when the overhead cost increases and the availability decreases from 0.5 to 0.2. When the overhead cost increases from 100 to 1000, the optimal number of time slots changes from 324 to 866. However, if the availability is equal to 0.4, the optimal number of time slots changes from 101 to 324 only. In other words, the optimal number of time slots is sensitive to the overhead cost if the availability is less than 0.6. However, if the availability is greater than 0.5, the optimal number of time slots is less sensitive to the change in overhead cost.

Figure 50 shows that when the overhead cost increases, the number of iterations decreases in general. Note that as the number of iterations increases, the overall scheduling cost increases too. As shown in Figure 50, the number of iterations is less than three when the overhead cost is greater than ten. However, the number of iterations increases from 3 to 79 when the overhead cost is equal to one and the availability decreases from 0.6 to 0.2. Furthermore, the optimal number of time slots tends to be more than 1000 time slots if the availability is less than 0.3. This large value is unrealized if we schedule a meeting in the real life. As a result, we can conclude that a meeting will be difficult to schedule if the availability of the invitees is too small, i.e the availability of all invitees is less than 0.3.

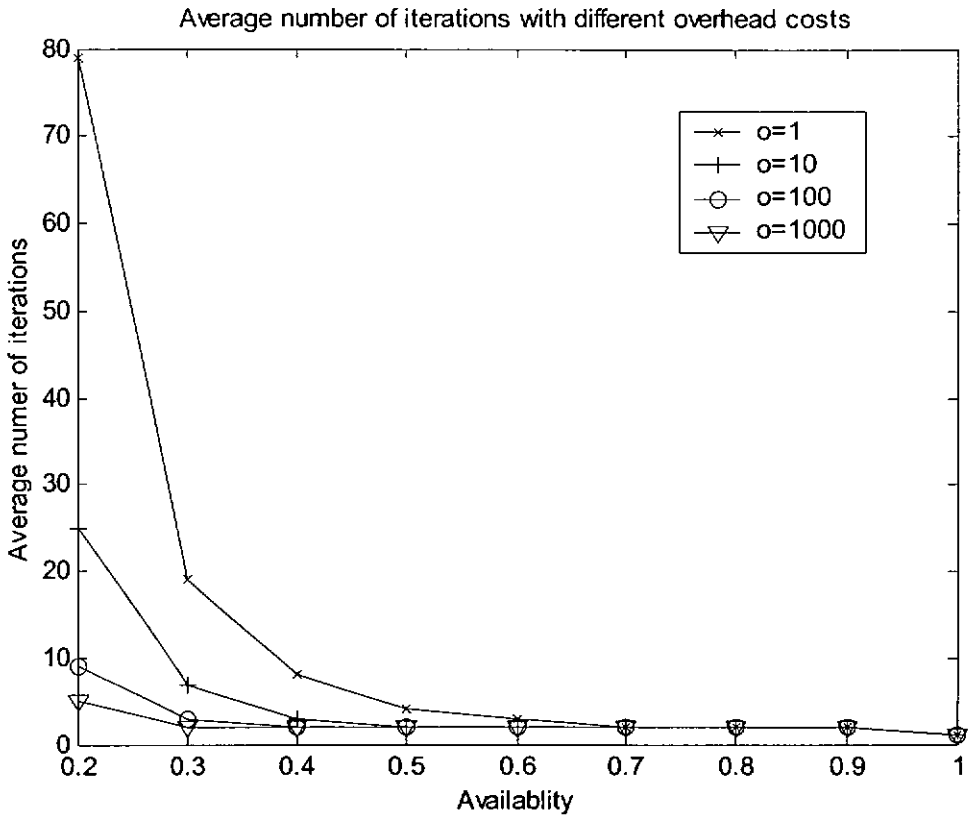

Figure 50. Average number of iterations with different overhead costs

From Figures 49 and 50, it can be seen that when the overhead cost increases and the availability decreases, the optimal number of time slots will increase and the average number of iterations will decrease. This is because when the overhead cost is high, it is better to provide more time slots in each round, in order to minimize the number of iterations and keep scheduling costs as low as possible. As a result, the average number of iterations is 2 when the overhead cost is equal to a thousand. However, if the overhead cost is equal to one, 19 iterations are needed when the availability is equal to 0.3, because only 35 time slots are provided to the invitees in each iteration.

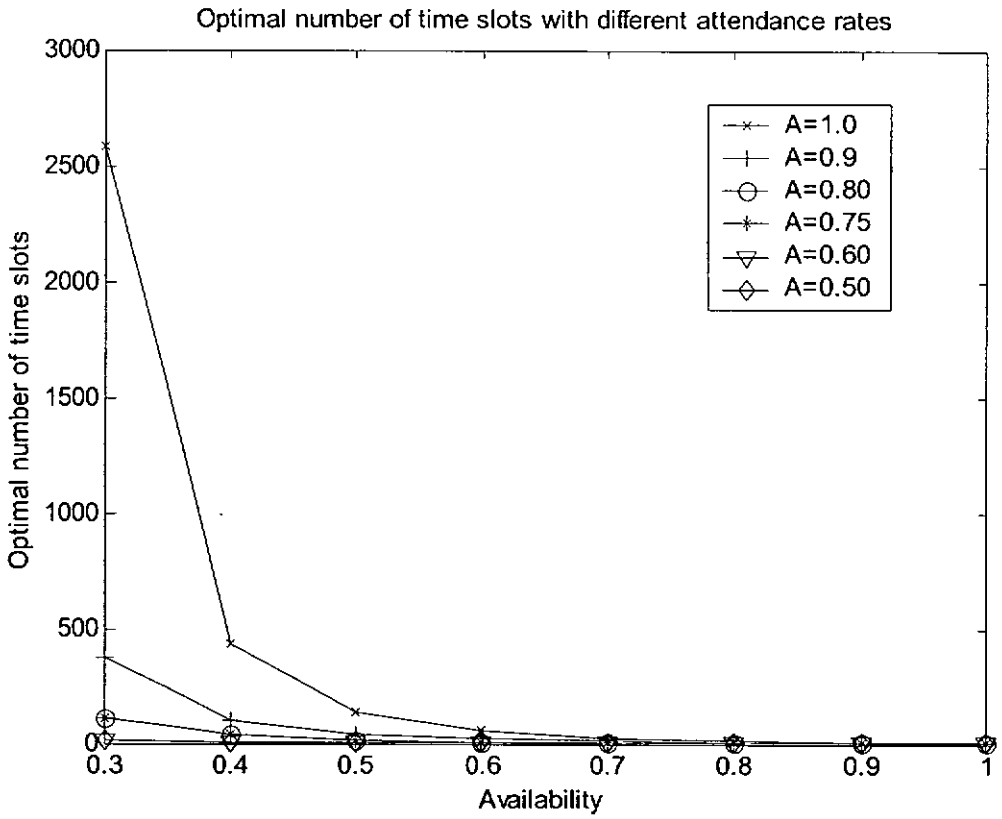

Figure 51. Optimal number of time slots with different attendance rates

As shown in Figure 51, the optimal number of time slots remains relatively constant when the availability is between 0.7 and 1.0. However, the optimal number of time slots increases sharply when the availability is less than 0.5 and the attendance rate is above 0.5. When the attendance rate increases, more participants are required to attend the meeting. The number of time slots required increases when more participants join the meeting, because the probability that a common time slot can be found decreases when the number of participants increases. As a result, the system needs to provide more time slots in each iteration. Furthermore, the number of time slots increases sharply when the attendance rate is high and the availability is low (less than 0.5). The optimal number of time slots needed for the 75% and 80% attendance rate is the same, because, due to the ceiling operation, the minimum number of attendants for the 75% and 80% attendance rate is the same.

Figure 52 shows the changes in the number of iterations when the attendance rate increases from 0.5 to 1.0. If the attendance rate is less than 0.8, the number of iterations required is less than three, irrespective of the availability. However, the number of iterations increases from 4 to 66 when all invitees are required to attend the meeting and their availabilities decrease from 0.6 to 0.3. When more invitees are required to attend the meeting, the system needs to propose more time slots in each iteration. In this case, both the number of iterations and the number of time slots for each iteration increase when the attendance rate is high.

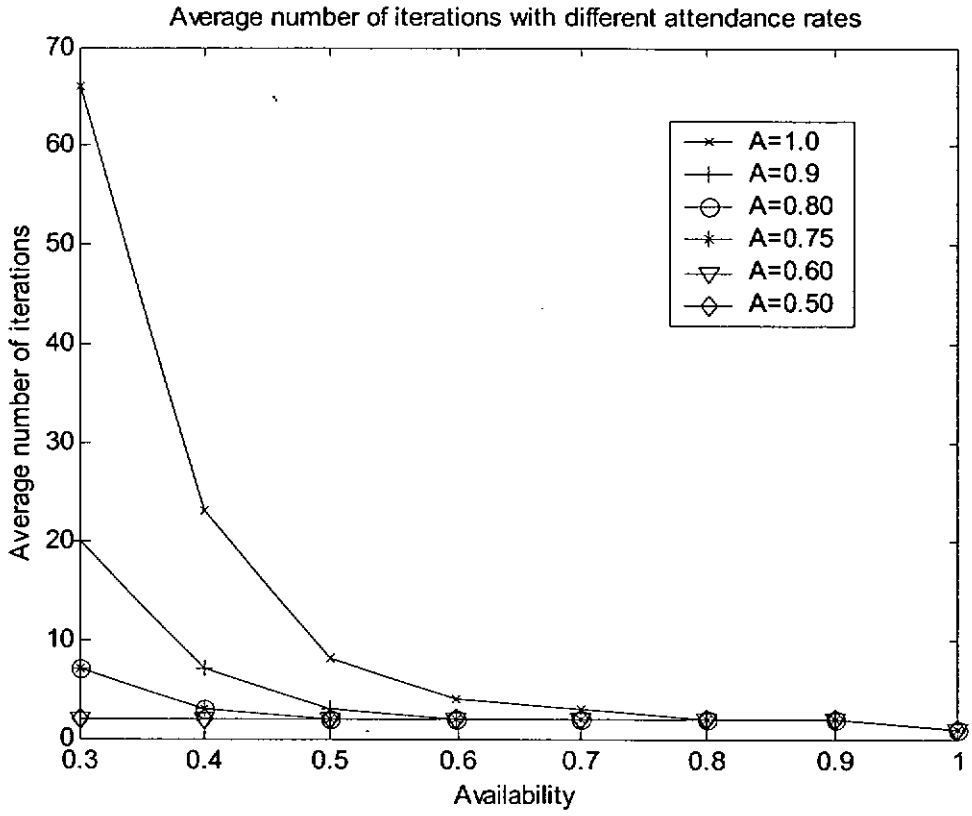

Figure 52. Number of iterations with different attendance rates

Based on the above results, it was found that the duration of the meeting has the greatest effect on the optimal number of time slots and the number of iterations required to schedule a meeting. Comparing the results as shown in Figures 47, 49 and 51, the number of time slots increases dramatically when the attendance rate increases

and the availability is less than 0.7. Also, Figures 48, 50 and 52 show that the number of iterations increases greatly when the duration of the meeting also increases. By contrast, the optimal number of time slots and the average number of iterations are insensitive to changes in the overhead cost and attendance rate when the availability is less than 0.5.

4.4 Summary

In this chapter, we have proposed a simple meeting scheduling mechanism and formulated a model to find the optimal number of time slots for scheduling a meeting such that the scheduling cost can be minimized. The scheduling cost depends on the number of time slots proposed to the invitees in each iteration and the number of iterations. As expected, when the length of the meeting, overhead cost or attendance rate of the meeting increases, the optimal number of time slots will also increase in general. However, it was found that the optimal number of time slots is insensitive to the duration of the meeting, the overhead cost of each iteration and the attendance rate if the availability is greater than 0.6. The analytical and simulation results indicate that if the availability is over 0.6, a good strategy should propose two or three time slots per iteration. Furthermore, on average, one or two iteration(s) are sufficient to schedule a meeting. This strategy is generally consistent with our experience in daily life.

CHAPTER 5
MARKOV DECISION MODEL FOR SCHEDULING

In summary, Figure 53 shows ASSIST's basic meeting scheduling mechanism, which is inspired by [14], [20]. Upon receiving a request, possible time slots are found based on information about the meeting. The system then sends these time slots to the invitees to enquire about their availability. To reflect the burden of filling in a time slot, there is a fill-in cost for each time slot. The system then waits for the replies. Note that when an invitee indicates that he/she is available at a particular time-slot, that time slot will be locked/reserved. To reflect this effect, there is a locking cost for each reserved time slot as the time slot cannot be used. Having received the replies, the system determines whether the available time slot(s) can be found for the meeting. If the result is successful, the system accordingly informs the people concerned. If not, a new scheduling cycle is initiated. In both cases, the unused time slots are unlocked. To start a new scheduling cycle, an overhead cost is incurred to the system. If the available time slot(s) cannot be found when the deadline is reached, there is a termination cost to indicate the adverse consequence. The aim of this section of the research is to investigate how many time slots should be proposed for scheduling a meeting in order to minimize the overall scheduling cost (see the later model). In particular, our model not only takes into account various costs but also the

scheduling deadline. As shown later, the solution suggests that the number of time slots should generally be changed dynamically (e.g., more time slots should be proposed as the scheduling deadline approaches).

In this chapter, we formulate a meeting scheduling problem based on a Markov decision model [22].

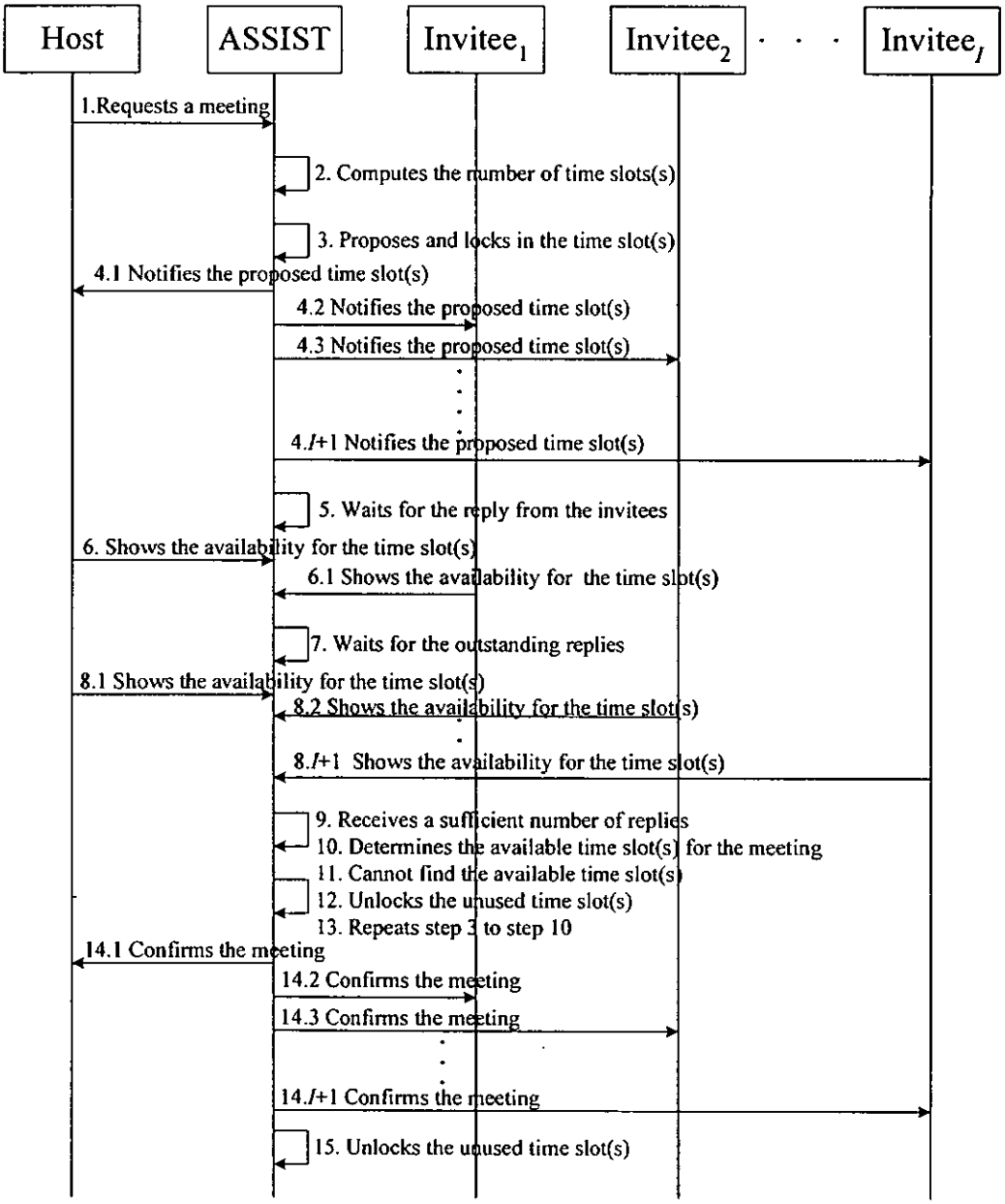

Figure 53. Meeting scheduling procedures

5.1 Markov Decision Model

As mentioned in the previous section, the system proposes one or more time slots and then waits for the replies from the invited people. At each decision point, the system checks the result. If all of the replies are received, it looks for the available time slot(s) for the meeting. If the result is unsuccessful, the system starts another scheduling cycle by proposing another set of time slot(s). The model, assumptions and notations used are given as follows.

To facilitate the analysis, we assume the following:

- o, l and f (as defined later) are constant throughout all of the scheduling cycles.
- All invitees have the same availability (i.e., the probability that an invitee is available at a time unit).

Table 11 below shows the notation used in the model.

Table 11. Notations used in the model

Symbol	Meaning
m	The number of the time slots proposed in each cycle
M	The maximum number of time slots proposed in each cycle
i	The i^{th} cycle
Q	Probability that the available time slot(s) can be found at the i^{th} cycle
f	Cost of filling in a time slot

o	Overhead cost of starting a new scheduling cycle
l	Cost of locking (i.e., reserving) a proposed time slot
C	Cost of failing to schedule a meeting
p	Probability that an invitee is available at a time unit
d	Required number of time unit(s) for a meeting
q	Probability that an invitee can attend the meeting (see later)
r	Probability of an invitee replying to the system (response rate)
A	Minimum attendance rate
I	Number of invitees
N	Minimum number of attendees
D	Scheduling deadline
n	Number of outstanding responses
R	Probability that the sufficient number of invitees reply to the system

Following the approach and similar notations used in [33], a Markov decision model is formulated in this section. A discrete time system is considered with decision points at 1, 2, 3, ... D, where D is the scheduling deadline. At each decision point, the state of the system changes. If the previous scheduling cycle fails, the system needs to start a new cycle by computing the number of time slots. There are three system states:

- Wait (W): the system is waiting for one or more outstanding replies.

- Fail (*F*): all of the replies are received but the available time slot(s) cannot be found.
- End (*E*): all of the replies are received and the available time slot(s) can be found.

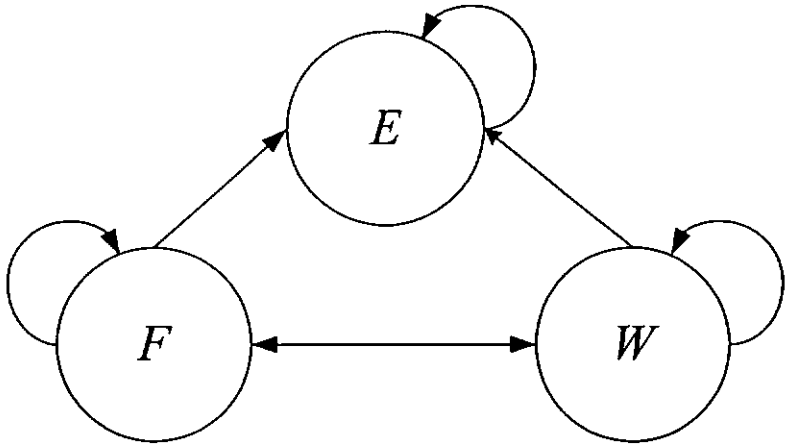

Figure 54. State diagram for meeting scheduling model

Figure 54 shows the corresponding state transition diagram. The system is in E state only if it receives a sufficient number of replies and a common time slot can be found. If the system waits for a reply from the invitees, it is in the W state. When the system receives a sufficient number of replies, it changes to F state or E state, depending on whether a free time slot can be found. If the system is in the F state, it must propose another set of time slots, and the system then changes to W state. It either changes to the E state if a common time slot can be found or changes back to the W state if the system waits for the replies from the invitees. Should the user's machine hang before the user receives the proposed time slots, users would not know the proposed time slots and the user would not be able to reply to the system. As a result, if the system receives a sufficient number of replies from the other invitees, it evaluates the results and finds a

common time slot, simply ignoring the availability of the user if there is no reply from the user. In other words, if a user's PC hangs after the user replies to the system, the system considers the availability of the user when it finds a common time slot.

The W state can be divided into sub-states $W_{m,n}$, where m is the number of proposed time slots for the current scheduling cycle and n is the number of outstanding responses. Note that n is between 1 and the minimum number of attendees for the meeting (N). For example, if there are four sub-waiting states and the system proposes three time slots; i.e., $W_{3,1}$, $W_{3,2}$, $W_{3,3}$ and $W_{3,4}$. Figure 55 shows the state transition for the four sub-waiting states. For instance, $W_{3,4}$ can be changed to other sub-waiting states but $W_{3,3}$ cannot change to $W_{3,4}$. When the system receives all of the replies, it can move to E state and F state if the available time slot(s) can and cannot be found, respectively. Obviously, once the E state is reached, no more state transitions will occur.

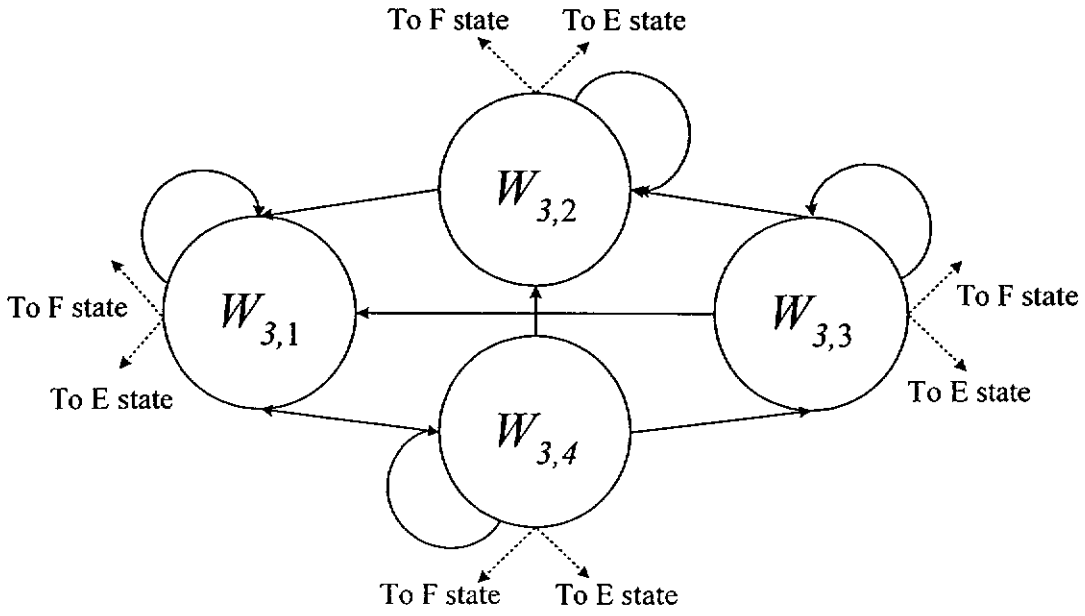

Figure 55. State diagram for waiting state with n=4

At the F state, the system needs to compute the number of proposed time slots for the next scheduling cycle; i.e., $m = \{1, 2, 3, \ldots M\}$.

Denote $T(\Omega_{t+1}|\Omega_t,\Theta_t)$ as the transition probability that the system state will change from Ω_t at t to Ω_{t+1} at $t+1$ if the action taken at t is:

$$\Theta_t = \begin{cases} m & \text{if } \Omega_t = F \\ null & \text{if } \Omega_t = W \text{ or } \Omega_t = E \end{cases} \tag{15}$$

Table 12 shows the transition probabilities where Q (more specifically, $Q(I, N, q, m)$) is the probability that the available time slot(s) can be found and R (more specifically, $R(I, N, n, r)$) is the probability that the outstanding replies are received. Q and R will be computed in the next sub-section.

Table 12. Transition probability

	$W_{m,n}$	F	E
$W_{m,n}$	Computed by (16)	R x (1-Q)	R x Q
F	Computed by (16) with $n=N$	R x (1-Q)	R x Q
E	0	0	1

Table 13 shows the scheduling cost at each state. It depends on the number of time slots proposed to invitees: if more time slots are proposed, the corresponding locking cost and fill-in cost is increased.

Table 13. Scheduling cost

	$W_{m,n'}$	F	E
$W_{m,n}$	I x l x m	I x l x m	I x l x m
F	$(I$ x l x $m)$ + $(I$ x f x $m)$ + o	$(I$ x l x $m)$ + $(I$ x f x $m)$ + o	$(I$ x l x $m)$ + $(I$ x f x $m)$ + o
E	0	0	0

5.1.1 Probability Q (probability that the available time slot(s) can be found)

As mentioned in the Chapter 4.2.2, the probability Q is calculated by the equation (13). In order to simplify the Markov Decision model for meeting scheduling, in chapter 5 we assume there is only one group of invitees.

5.1.2 Probability R (the probability that the sufficient number of invitees reply to the system)

According to Table 11, an outstanding invitee replies to the system with probability r at a decision point. It is not difficult to see that the transition probability from $W_{m,n}$ to $W_{m,n'}$ is:

$$T\left(W_{m,n'}\middle|W_{m,n}\right)=\binom{I-(N-n)}{n-n'}r^{n-n'}(1-r)^{I-N+n'} \qquad (16)$$

where $n \geq n'$. Recall that the action is null at the waiting state. If there are n outstanding replies, the probability that the system receives sufficient number of replies from the invitees is

$$R(I,N,n,r)=1-\sum_{j=0}^{n}\binom{I-(N-j)}{n-j}r^{n-j}(1-r)^{I-N+j} \qquad (17)$$

5.1.3 Backward induction method to find the optimal number of proposed time slots

Having defined the above Markov decision model, we use the backward induction method to find the optimal policy as well as the minimum scheduling cost (see [33] for details on the backward induction method). Denote $Z_{t+1}^*(F)$ and $Z_{t+1}^*(W_{m,n})$ as the minimal average scheduling cost from $t+1$ to D when the system state is F and $W_{m,n}$, respectively. This means that the optimal number of time slots to be proposed at $t+1$ is the one which results $Z_{t+1}^*(F)$. Suppose that at t, the system proposes/has proposed m time slots and the number of outstanding replies is n if applicable. Under the above condition at t, let $Z_t(F,m)$ and $Z_t(W_{m,n})$ be the following:

$$Z_t(F,m)=\delta_t(F,m)+T(F|F,m)\times Z_{t+1}^*(F)+\sum_{n=1}^{N}T(W_{m,n}|F,m)\times Z_{t+1}^*(W_{m,n}) \qquad (18)$$

$$Z_t(W_{m,n})=\delta_t(W_{m,n})+T(F|W_{m,n})\times Z_{t+1}^*(F)+\sum_{n'=1}^{n}T(W_{m,n'}|W_{m,n})\times Z_{t+1}^*(W_{m,n'}) \qquad (19)$$

where $\delta_t(F,m)$ and $\delta_t(W_{m,n})$ represent the immediate or incremental cost at t as defined below:

$$\delta_t(F,m)=I\times m\times f+I\times m\times l+o \qquad (20)$$

$$\delta_t(W_{m,n})=I\times m\times l \qquad (21)$$

Based on (18)-(21), $Z_t^*(F)$ can then be found as follows:

$$Z_t^*(F) = \min\{Z_t(F,1), Z_t(F,2), \ldots, Z_t(F,M)\} \quad (22)$$

where $\min\{.\}$ denotes the smallest term inside the bracket.

This means that if $Z_{t+1}^*(F)$ is known, $Z_t^*(F)$ can also be found. At the last decision point (the deadline), the termination cost is C if the available time slot(s) cannot be found (i.e., not at the state E). Therefore we have

$$Z_D^*(F) = C \quad (23)$$

$$Z_D^*(W_{m,n}) = C \quad (24)$$

Starting from (23) and (24), we can work backward (i.e., D, D-1, D-2,...,1) to obtain the optimal policy (i.e., the optimal number of time slots to be proposed at state F at each decision point) based on (18)-(21).

5.2 Analytical Results and Discussions

Based on the above model, this section presents and discusses some analytical results. Table 14 shows the base parameters used in the following analysis.

Table 14. Value of the parameters used in the model

Parameter	Meaning	Value
I	Number of invitees	10
f	Cost of filling in one time slot	1
o	Overhead cost of each scheduling cycle	10
l	Cost of locking one time slot	1
C	Termination cost of failing to schedule the meeting	100,000
p	Availability (probability that a person is available at a time unit)	0.8
r	Response rate of an invitee	0.8
d	Duration of the meeting (time unit(s))	1
A	Minimum attendance rate	75%
M	Maximum number of time slots	20
D	Scheduling deadline	7

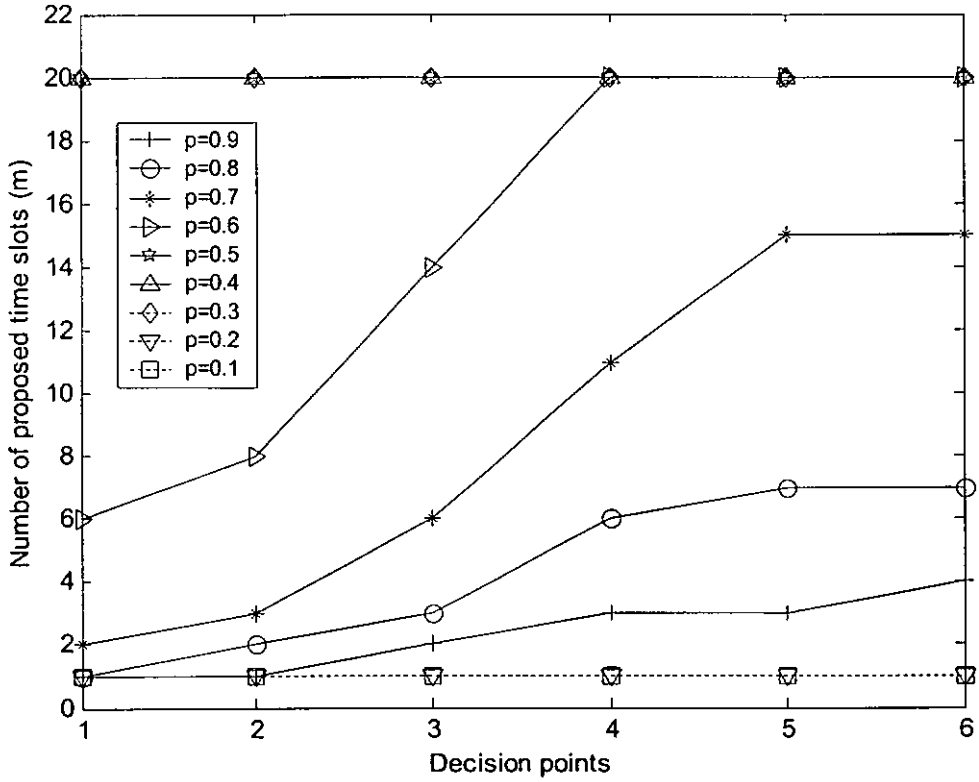

Figure 56. Number of time slots with different availabilities

Figure 56 shows that, in general, the number of time slots increases as the deadline approaches. If the availability is smaller, the increase is larger. The number of the proposed time slots increases greatly from 6 to 20 when the availability is 0.6. In addition, the number of time slots remains at 20 (the maximum number of proposed time slots) if the availability is between 0.4 and 0.5. However, the number of time slots remains at 1 if the availability is less than 0.4. This indicates that the available time slot(s) cannot be found if the availability is too low. In this case, the system only provides a very small number of time slots in order to minimize the scheduling cost.

Figure 57 shows that if the response rate is small or large, the number of proposed time slots decreases or increases, respectively as the deadline approaches. This is because if the response rate is low, it is better to propose

more time slots at the beginning so that the system has a better chance of receiving the responses. On the other hand, if the response rate is high, it is better to increase the number of time slots gradually in order to minimize the scheduling cost.

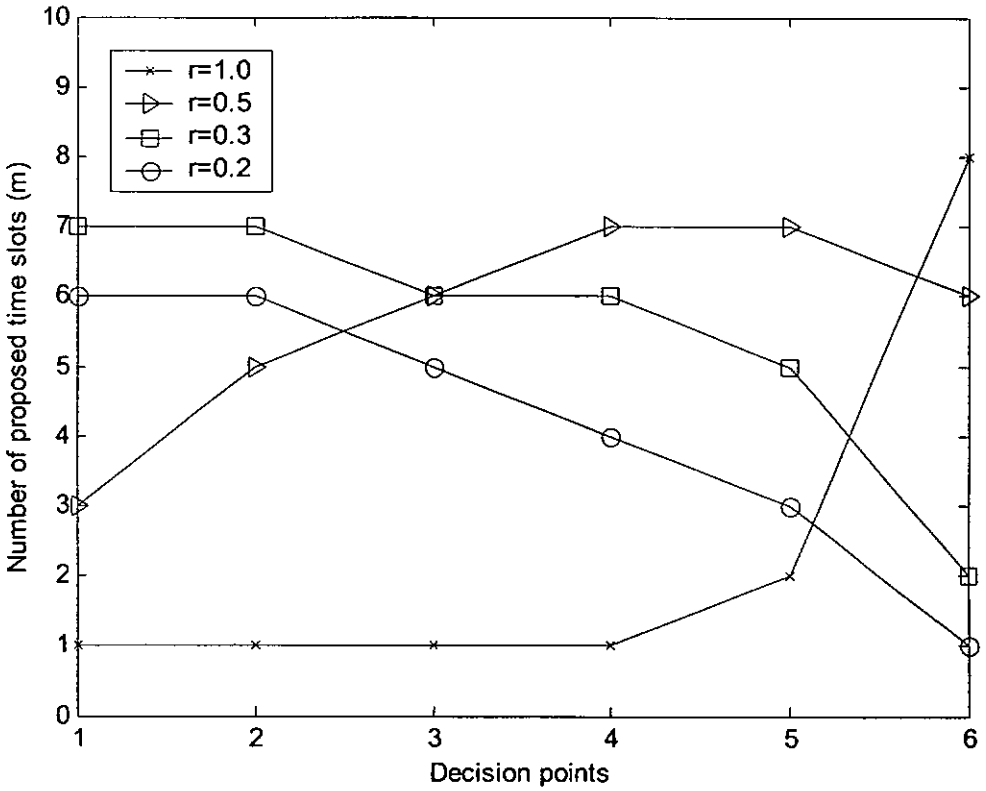

Figure 57. Number of time slots with different response rates

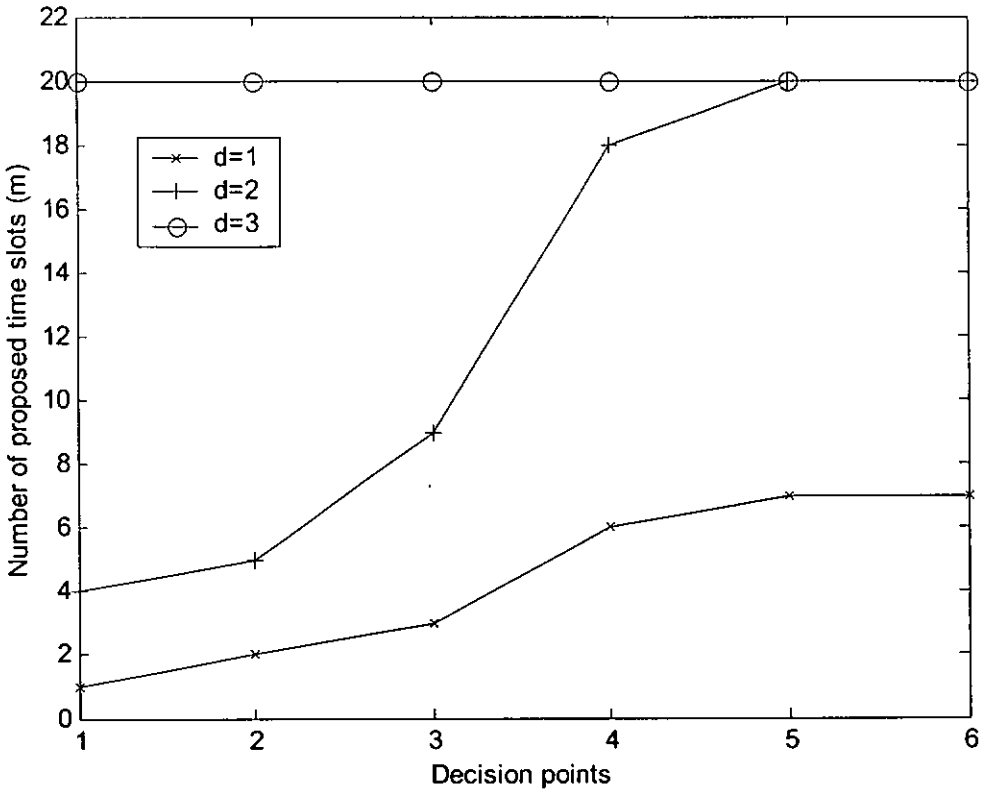

Figure 58. Number of time slots with different durations

Figure 58 shows how the duration of the meeting affects the number of proposed time slots at each scheduling cycle. Obviously, when the duration of the meeting increases, the number of proposed time slots should also increase. It can also be seen that when the duration of the meeting is very long (d=3), it is better to propose the maximum number of time slots at all decision points.

Figure 59 shows the effect of the overhead cost of starting a new scheduling cycle. It can be seen that the overhead cost has a greater effect at the earlier decision points as expected. When approaching the deadline, the four curves converge. In particular, the number of proposed time slots remains at seven at the last two decision points.

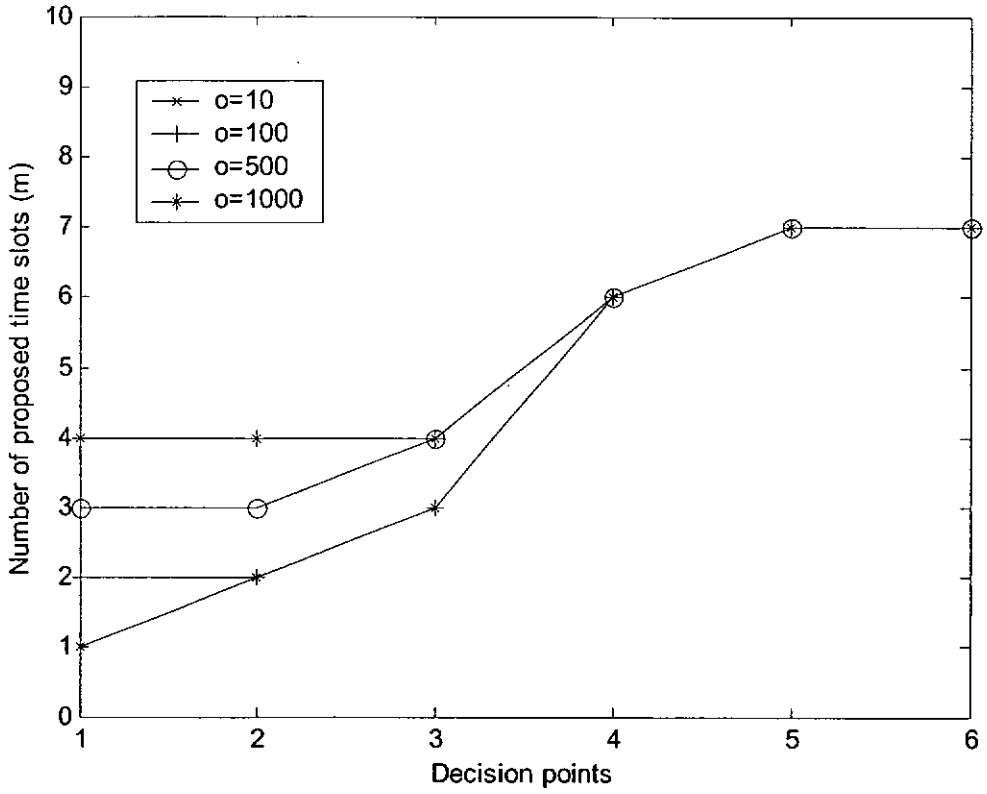

Figure 59. Number of time slots with different overhead costs

Figure 60 shows the effect of the filling cost and locking cost. Note that when the number of proposed time slots increases, the overall costs for filling in and locking in the time slots also increase. As a result, the required number of proposed time slots should decrease when the filling cost or locking cost increases, in order to keep the scheduling cost as small as possible.

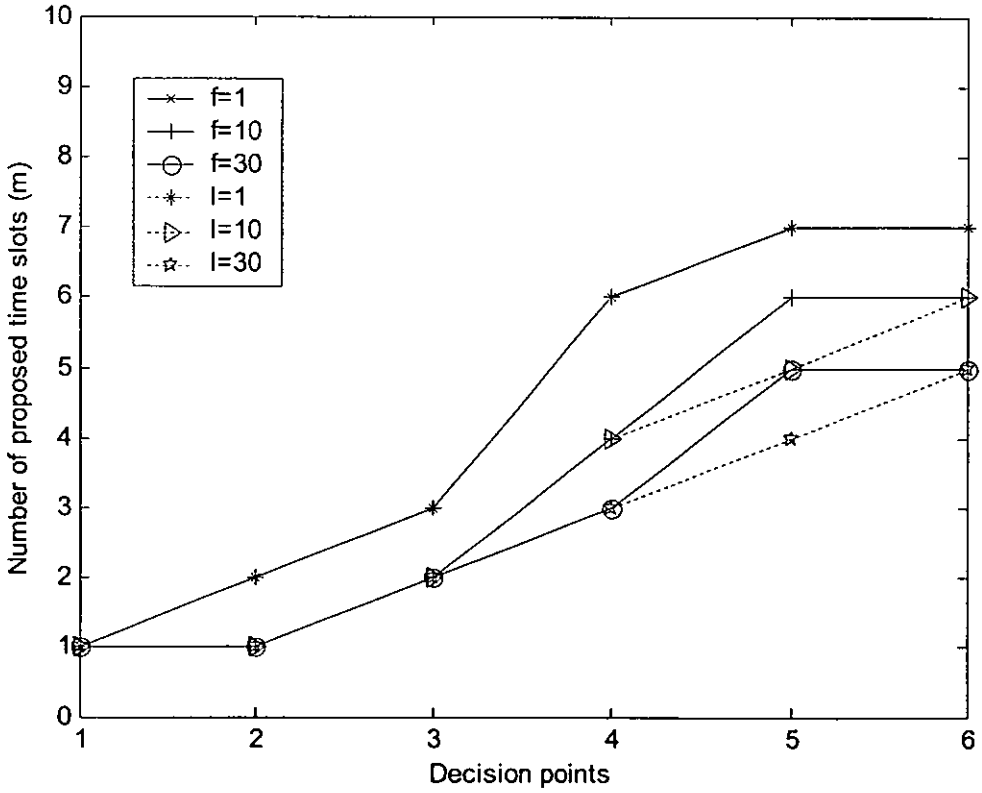

Figure 60. Number of time slots with different filling costs/locking costs

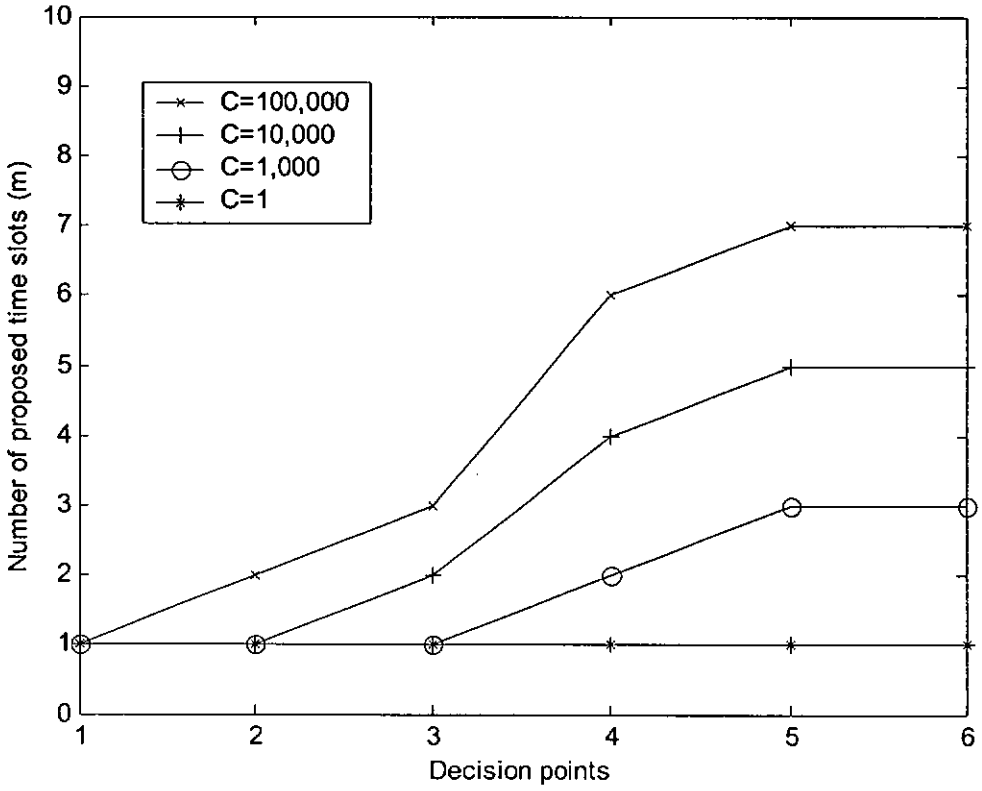

Figure 61. Number of time slots with different terminating costs

Figure 61 shows the effect of the termination cost, which reflects the importance of the meeting. In other words, a very important meeting should have a large termination cost to indicate the adverse consequence of failing to schedule the meeting. If the termination cost is low, it is better to keep proposing a smaller number of time slots, so as to minimize the scheduling cost. If the termination cost is large, more time slots should be proposed, especially if the deadline is approaching.

Figure 62 shows that more proposed time slots should be proposed when the attendance rate is larger. If the attendance rate is close to 100%, the maximum number of time slots should be proposed at each decision point.

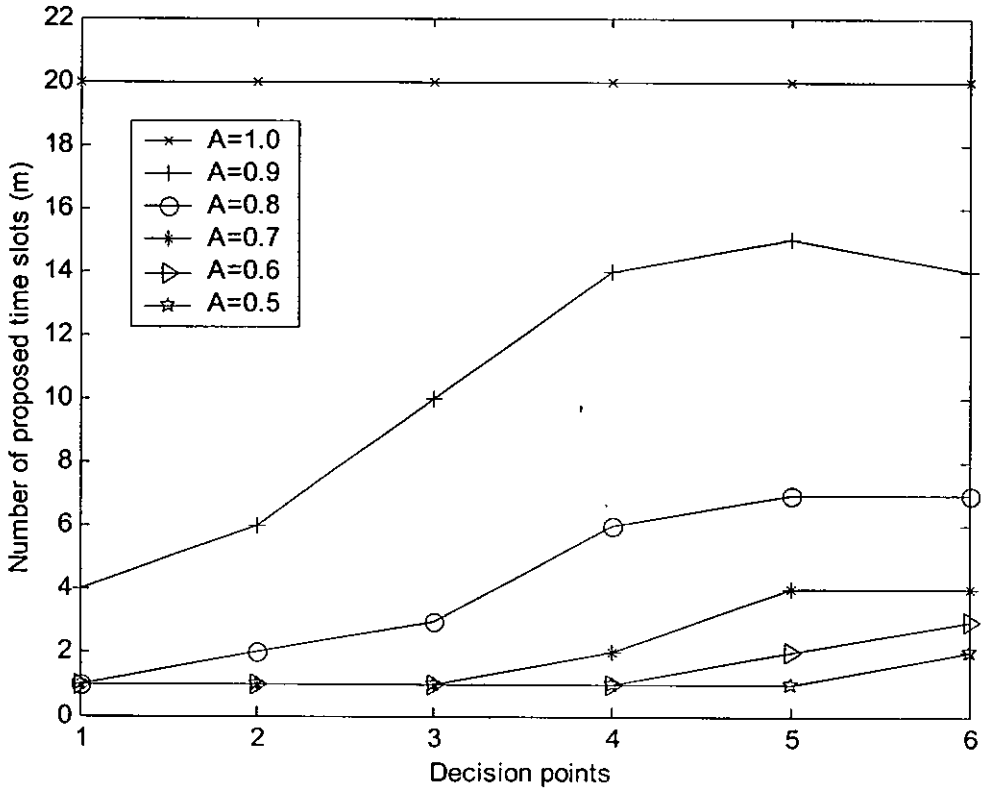

Figure 62. Number of time slots with different attendance rates

Figure 63 shows the effect of the maximum number time slot number (M). As expected, it indicates that the maximum number of time slots has a greater effect if the availability is low because, in this case, a larger number of time slots are required. The solid lines show the optimal number of proposed time slots if the availability is equal to 0.8. The optimal number of proposed time slots is seven at decision points 5 and 6. If the maximum number of time slots is greater than five, the number of proposed time slots with the availability 0.8 is not affected by the maximum limit on the proposed time slots. As a result, the number of proposed time slots remains at around the same level even if the maximum number of time slots is increased. However, the number of proposed time slots remains at M if M is less than 30 and the availability is equal to 0.6. For example, the number of proposed time slots is changed from 20 to 25 at decision point 4 when the maximum number of time slots is increased from 20 to 30. The system only proposes the maximum number of time slots if the optimal number of time slots is greater than the limit. Note that the maximum number of time slots restricts the number of proposed time slots in each scheduling cycle.

Figure 64 shows the optimal number of time slots for different availabilities if there is no limit on the number of proposed time slots. It can be seen that significantly more time slots should be proposed if the availability is low. In this case, it is very difficult to schedule a meeting unless the invitees are willing to fill in a large number of time slots.

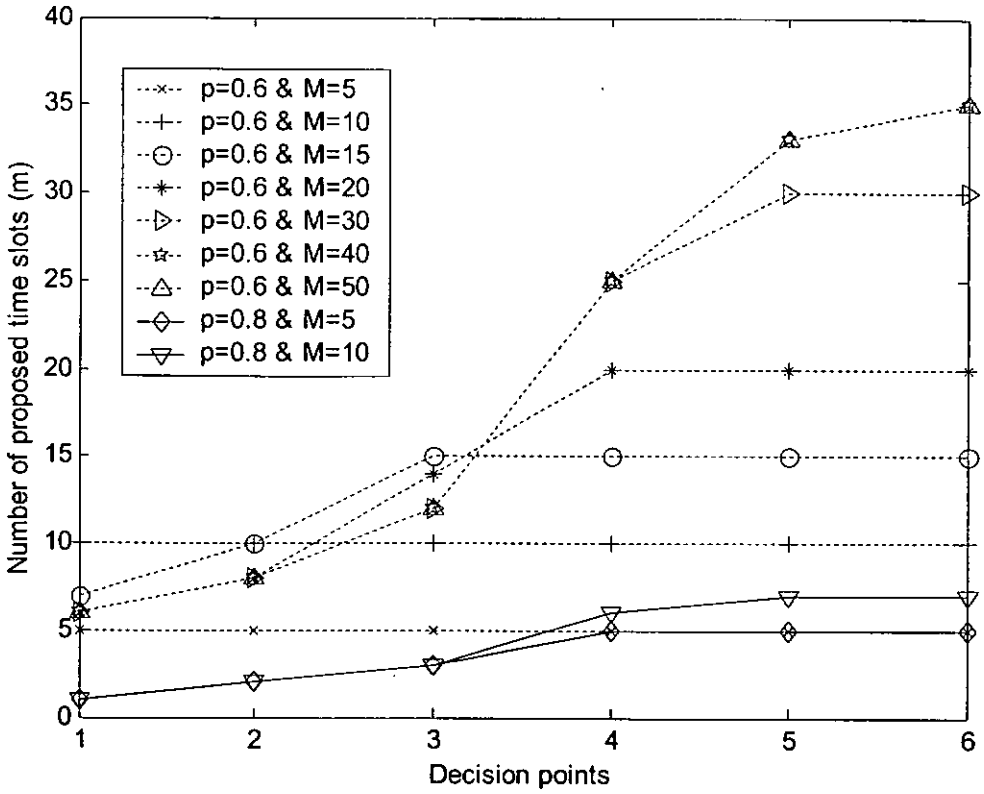

Figure 63. Number of proposed time slots with different maximum number of time slots

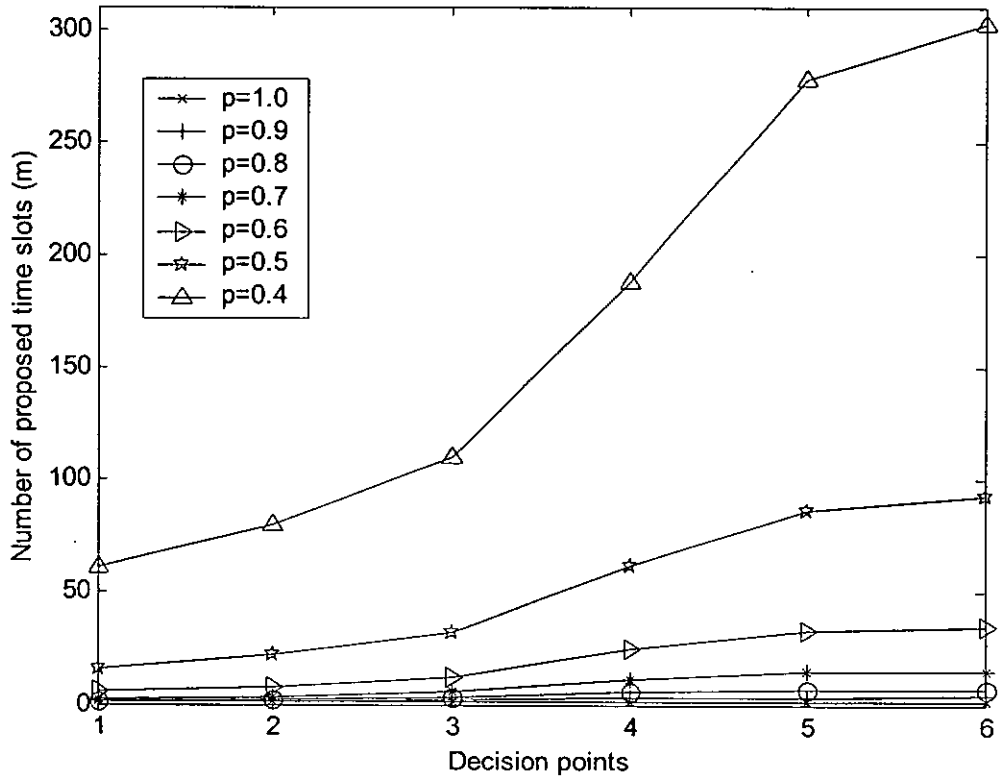

Figure 64. Optimal time slot with different availabilities

As shown in Figure 64, the optimal number of time slots increases when the availability decreases. Furthermore, the optimal number of time slots increases when the schedule is close to deadline. The optimal number of time slots when the availability is equal to 0.4 is much higher than the optimal number of time slots when availability is higher than 0.5. The optimal number of time slots is realistic if the availability is greater than 0.7, because the optimal number of time slots can be less than 10 at each decision point.

The above model suggests that in order to keep the scheduling cost at a minimum, the number of time slots proposed at each decision point should be varied dynamically. It is of interest to compare this dynamic time slot policy with a fixed time slot policy. In the latter case, a fixed number of time slots (m) are used at all decision points. Figure 65 shows the difference in cost between the two approaches. It can be seen that the cost difference is more significant when the availability is between 0.3 and 0.7. This is because when the availability is high, it is better to propose a small number of time slots. When the availability is low, it is difficult to schedule the meeting, irrespective of the policy used. In other words, the termination cost is likely to be incurred.

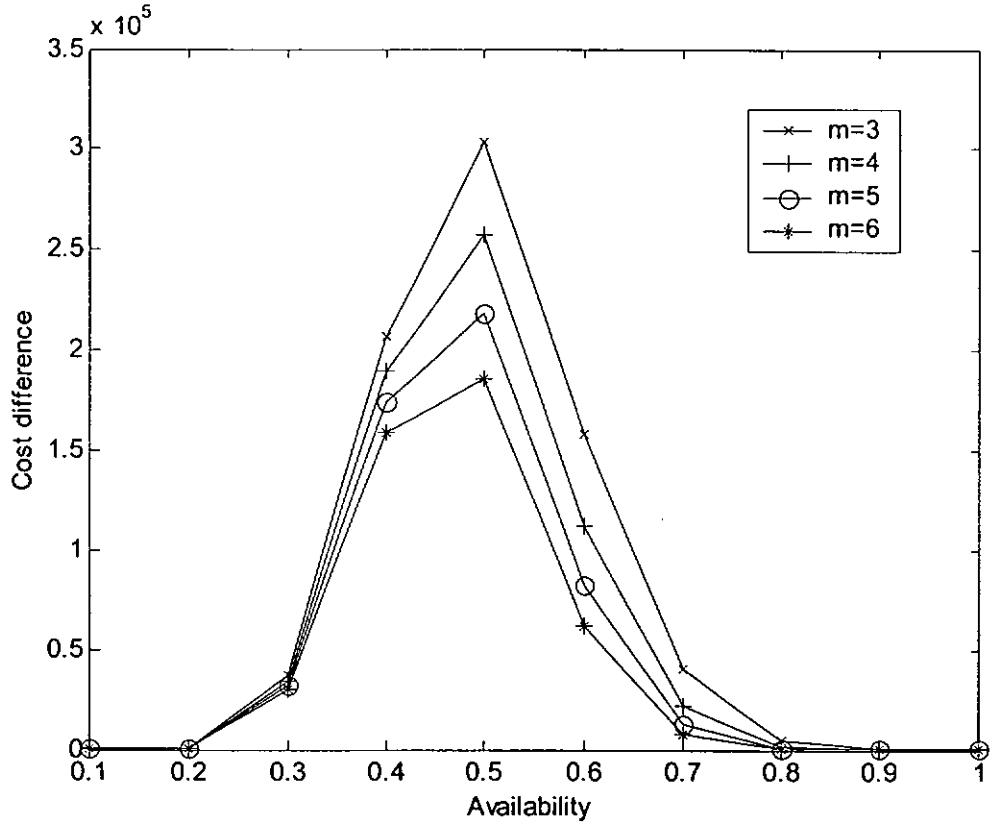

Figure 65. Cost comparison between the dynamic and fixed time slot strategy

5.3 Summary

We have proposed a meeting scheduling mechanism based on a Markov decision model. The model takes into account various costs and the scheduling deadline. By using the backward induction algorithm, we have obtained the optimal policy (i.e., the optimal number of time slots to be proposed at each decision point for scheduling purposes in order to minimize the overall expected scheduling cost). Analytical results have been presented to show how the system's behavior is affected by various parameters: meeting duration, overhead cost, locking cost, fill-in cost or attendance rate. The results should give valuable insights into designing meeting scheduling systems in general and automatic secretaries in particular.

CHAPTER 6
MATCHING ALGORITHMS

In chapter 4 and 5, we proposed models for finding the optimal number of proposed time slots. In this chapter, we propose several methods for finding a common time slot within the proposed time slots. In order to find a common time slot for a meeting from the proposed time slots, we investigate 5 different matching approaches. They are the random matching algorithm (RMA), ascending-availability matching algorithm (AAMA), descending-availability matching algorithm (DAMA). Also we investigate a modified ascending availability matching algorithm and descending availability matching algorithm. By classification, the meeting algorithms can be divided into sorting or non-sorting approaches based on the availability of the replies from the invitees. The diagram below shows the classification of the matching algorithms.

6.1 Matching Models

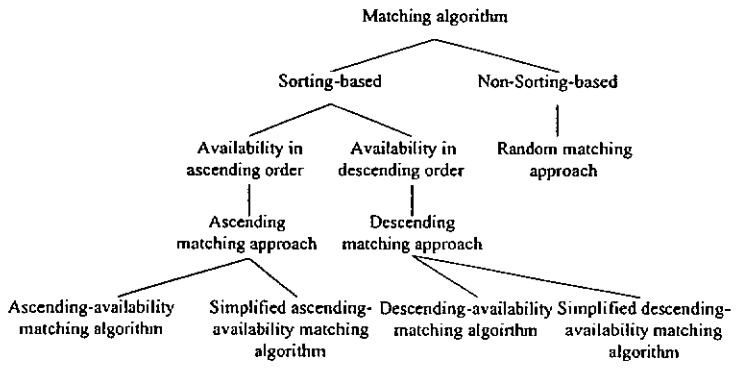

Figure 66. Number of iterations with different attendance rates

As shown in Figure 66, the five matching algorithms can be divided into sorting and non-sorting approaches. Random matching belongs to the non-sorting based approach, because no preprocessing on the matching order of the invitees is required. The sorting-based algorithm rearranges the matching order based on the ascending or descending order in the availability of the repliers, such as ascending availability matching algorithm and descending availability matching algorithm.

1. Non-sorting matching approach

    a. Random matching algorithm (RMM)

        Random matching approach is the non-sorting matching approach and no sorting on the matching order of the invitees is required. It identifies a common time slot among the invitees based on the order of the replies. The figure below shows the random matching algorithm.

Random Matching Algorithm

1. Check the availability of the first invitee at the earliest un-checked time slot
2. Repeat step 1 by checking the availability of the next invitee at the same time slot
3. Stop checking
   a. the requirement of the minimum number of the attendants is met or
   b. insufficient number of attendants is available at the time slot
4. If the time slot cannot be assigned to the meeting, then
   a. unlock the time slot and
   b. repeat Step 1
5. Stop the matching processing when
   a. a common time slot is assigned to the meeting
   b. no common time slot can be found

Figure 67. Random matching algorithm

Case study

The following example demonstrates how the random matching algorithm finds out the common time slot among six invitees (A to F). Four time slots (I to IV) are the proposed to the invitee A to F. Figure 68 shows the availabilities of the invitees from the time slots I to IV. "✓" means that the invitee is available at a particular time slot, while "×" means that the invitee is unavailable at the time slot. "?" indicates the system does not know the availability of the invitee at the proposed time slots, because the invitee does not reply to the system. The meeting

can be scheduled successfully only if 50% of the invitees can attend the meeting, i.e., any 3 of the invitees can attend the meeting. Furthermore, user A is the first user who replies to the system, followed by user B, C, D and E.

Invitee / Time slot	A	B	C	D	E	F
I	✓	×	×	×	✓	?
II	✓	×	×	×	✓	?
III	✓	✓	×	×	✓	?
IV	×	✓	×	✓	✓	?

Figure 68. Number of iterations with different attendance rates

As shown in Figure 68, invitee F does not reply the system. Invitee E is available at all the proposed time slots, while invitee C is unavailable at all the proposed time slots. Using the random matching algorithm, no sorting on the availability of the repliers is required. Figure 69 shows the checking sequence if the random matching algorithm is used.

Invitee / Time slot	A	B	C	D	E	F
I	✓ → ×	→ ×	→ ×		✓	?
II	✓ → ×	→ ×	→ ×		✓	?
III	✓ → ✓	→ ×	→ ×		→ ✓	?
IV	×	✓	×	✓	✓	?

Figure 69. Checking order based on the random matching algorithm

The random matching algorithm checks the availability of invitee A at the earliest unchecked time slot (i.e., Time slot I), and then B and so on. When the algorithm checks the time slot I, it stops after checking the availability of the invitee D as shown in Figure 69 because only user A is available at time slot I. Hence the time slot I must not be a common time slot, no matter invitee E is available or not. As a result, the matching

algorithm starts a next cycle to check the availability of invitee A at time slot II. The algorithm first checks the availability of invitee A, B, C and then D. It stops after checking the availability of invitee D again, because only invitee A is available at the time slot II after checking the invitee D, but the meeting requires at least 3 invitees to attend the meeting. Again, the algorithm starts a new cycle to check the time slot III. In the new cycle, the system records that invitee A and B are available at time slot III, but invitee C and D are unavailable at the same time slot. After checking the availability of invitee E, the common time slot is found because invitee E is available at time slot III too. As a result, at least half of the invitees are available at time slot III for attending the meeting, although invitee B and C are not available and invitee F may be available at time slot III.

Based on the above example, the system needs to check 3 time slots and uses 13 checkings to find the common time slot for the meeting. Note that 4 checkings for the time slot I and time slot II, and 5 checkings are spent for the time slot III.

2. Sorting matching approach

As shown in Figure 66, the sorting-based matching algorithm can be divided into the ascending and descending order on the availability of the invitees. The algorithm sorts the checking order based on the availability of the invitees before it starts to find the common time slot.

a. Ascending-availability matching algorithm

The ascending-availability matching algorithm sorts the checking order based on the availability of the invitees. The sorting algorithm rearranges the order of the invitees from the one who has the least available time slots to the one who has the most available time slots. Figure 70 shows the checking order of the invitees after the sorting based on the availability.

Invitee / Time slot	C	D	B	A	E	F
i	×	×	×	✓	✓	?
ii	×	×	×	✓	✓	?
iii	×	×	✓	✓	✓	?
iv	×	✓	✓	×	✓	?

Figure 70. Ascending order on the availability of the invitees

Invitee C is the busiest person among all invitees, because he/she cannot attend the meeting at the all proposed time slots, however, invitee E has the highest availability because he/she can attend the meeting at any proposed time slots. After sorting the checking order based on the availability of the invitees, the system checks the invitee C first, and then invitee D, B, A and finally E. Invitee F does not show the availability on the proposed time slot, the system cannot check the availability of F at the proposed time slot. The diagram below shows the detailed algorithm of the ascending-availability matching algorithm.

Ascending-availability matching algorithm

1. Sort the checking order based on the availability of the invitees. The busiest invitee is listed first.
2. Check the availability of the first invitee at the earliest un-checked time slot.
3. Repeat step 1 by checking the availability of the next invitee at the same time slot.
4. Stop checking if
   a. the requirement of the minimum number of attendants is met or
   b. insufficient number of attendants
5. If the time slot cannot be assigned to the meeting, then
   a. unlock the time slot and
   b. repeat Step 1
6. Stop the matching processing when
   a. a common time slot is assigned to the meeting
   b. no common time slot can be found

Figure 71. Algorithm of the ascending-availability matching algorithm

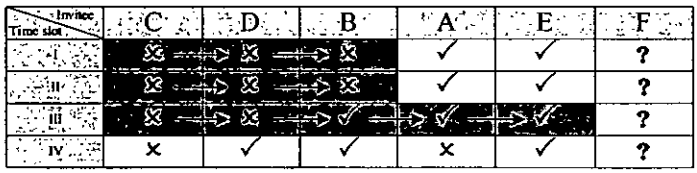

Figure 72. Checking sequence using ascending-availability matching algorithm

Figure 72 shows the checking sequence if the ascending-availability matching algorithm is used. When the system checks the first time slot (time slot I), it checks the availability of invitee C, D and finally B. As shown in

Figure 72, invitee C, D and B are not available at the time slot I, the system does not need to check the available status of invitees A and E at the time slot I. This is because the minimum attendance rate cannot be reached, even if invitees A and E are available at time slot I. Having checked the time slot II, the system starts a new cycle to check the availability of the invitees at time slot III. The meeting can be scheduled at the time slot III, because invitees A, B and E are available at time slot III.

Based on the above example, time slot III is assigned to the meeting, because all three invitees can attend the meeting at the time slot. 3 checkings have been performed on the time slot I and II. 5 checkings are needed on the time slot in order to find the common time slot.

b. Descending-availability matching algorithm

Except the random matching method, all of the matching algorithms rearrange the checking order of the invitees according to their availabilities. Descending-availability matching algorithm is similar to the ascending-availability matching algorithm, except the order of sorting. Descending-availability matching algorithm rearranges the checking order from the invitee with the most available time slots first. Figure 73 shows the detailed algorithm on the descending-availability matching algorithm and Figure 74 shows the checking order after the sorting process.

Descending-availability matching algorithm

1. Sort the checking order based on the availability of the invitees. The most available invitee is listed first.
2. Check the availability of the first invitee at the earliest un-checked time slot.
3. Repeat step 1 by checking the availability of the next invitee at the same time slot
4. Stop checking if
   a. the minimum number of attendants is reached or
   b. there are insufficient number of attendants
5. If the time slot cannot be assigned to the meeting, then
6. Unlock the time slot and
7. Repeat Step 1
8. Stop the matching processing when
   a. a common time slot is assigned to the meeting
   b. no common time slot can be found

Figure 73. Algorithm of the descending-availability matching algorithm

Invitee / Time slot	E	A	B	D	C	F
I	✓	✓	×	×	×	?
II	✓	✓	×	×	×	?
III	✓	✓	✓	×	×	?
IV	✓	×	✓	✓	×	?

Figure 74. Descending order on the availability of the invitees

The system checks the availability of invitees E, A, B, D and finally C. Invitee E is the most available person because he/she can attend the meeting at any proposed time slot. However, invitee C is the busiest person because he/she is unavailable at all the proposed time slots. Figure 75 shows the

checking sequence if using the descending-availability matching algorithm. Time slot III is assigned to the meeting, because invitees A, B and E can attend the meeting at the same time slot. The descending-availability matching algorithm spends 5 checkings for the time slot I and II, because only two invitees can attend the meeting. Moreover, only 3 checkings are required for checking time slot III.

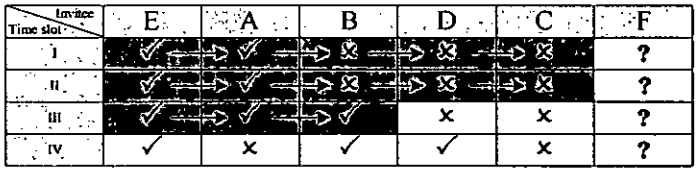

Figure 75. Checking order when using descending-availability matching algorithm

Simplification on the Ascending-availability matching algorithm & Descending-availability matching algorithm

Based on the ascending-availability matching algorithm and the descending-availability matching algorithm, we have proposed two more matching algorithms. They are similar to the ascending-availability and the descending-availability matching algorithm, but with further enhancements or simplifications. The simplified ascending-availability and the simplified descending-availability matching algorithm start a new cycle to check the next time slot when any one of the invitees is unavailable at the time slot. Figure 76 shows the matching sequence when using the simplified ascending-availability matching algorithm and Figure 77 shows the matching sequence when using the simplified descending-availability matching algorithm.

a. Simplified Ascending-availability matching algorithm

Time slot \ Invitee	C	D	B	A	E	F
I	X	×	×	✓	✓	?
II	X	×	×	✓	✓	?
III	X	×	✓	✓	✓	?
IV	X	✓	✓	×	✓	?

Figure 76. Checking order when using the simplified ascending-availability matching algorithm

As shown in Figure 76, the system checks the availability of the invitee C at the first time slot (time slot I), and then it starts to check the availability of the invitee C at the next time slot (time slot II) rather than the availability of invitee D at time slot I, because invitee C is unavailable at time slot I. As a result, the system checks the availability of invitee C at time slot II. The algorithm stops until it finds a common time slots among the invitees or it has checked all time slots but cannot find a common time slot. According to the above example, no common time slot can be found, because invitee C is unavailable at all of the proposed time slots.

b. Simplified Descending-availability matching algorithm

Figure 77 shows the checking sequence using the simplified descending-availability matching algorithm. During the first and the second cycle, the system checks the next time slot after checking the availability of invitee B, because B is unavailable at time slots I and II. Finally, time slot III is assigned to the meeting, because the first three invitees (A, B and E) are available to attend the meeting.

Time slot \ Invitee	E	A	B	D	C	F
I	✓	✓	✗	✗	✗	?
II	✓	✓	✗	✗	✗	?
III	✓	✓	✓	✗	✗	?
IV	✓	✗	✓	✓	✗	?

Figure 77. Checking order when using the simplified descending-availability matching algorithm

Comparison between the matching algorithms

We have introduced five matching algorithms in the previous section, they are random matching algorithm, ascending-availability matching algorithm, simplified ascending-availability matching algorithm, descending-availability matching algorithm and simplified descending-availability matching algorithm. Table 15 below compares their characteristics based on the previous example.

Table 15. Comparison on the characteristic of different matching algorithms

Features	**Random matching algorithm**	**Ascending-availability matching algorithm**	**Simplified ascending-availability matching algorithm**	**Descending-availability matching algorithm**	**Simplified descending-availability matching algorithm**
The meeting can be scheduled successfully	Yes	Yes	No	Yes	Yes
Meeting time	Time slot III	Time slot III	Nil	Time slot III	Time slot III
Total number of checkings	13	11	4	13	9
Total no of checking in the 1st cycle	4	3	1	5	3
Total no of checking in	4	3	1	5	3

the 2nd cycle					
Total no of checking in the 3rd cycle	5	5	1	3	3
Total no of checking in the 4th cycle	Nil	Nil	1	Nil	Nil

As shown in Table 15, only the simplified ascending-availability matching algorithm cannot find a common time slot among the invitees for the meeting. Although the other 4 matching algorithms can assign a common time slot for a meeting (i.e., time slot III), the number of checkings of each algorithm varies. Based on the above example, the simplified descending-availability matching algorithm performs better than the others, because only 9 checkings are needed. Note that the simplified ascending-availability matching algorithm cannot find a time slot of the meeting, because the simplified ascending-availability matching algorithm checks the availability of the busiest invitee first. When the user is unavailable at the time slot, the system starts to check the next time slot and so on. However, the busiest invitee (i.e. invitee C) is unavailability at all time slots, so the system cannot find a common time slot when using the simplified ascending-availability matching algorithm.

In the above examples, the matching algorithms stop when the first common time slot is found. If the aim of the algorithm is to find a common time slot with the maximum attendance, then the algorithm should check the attendance rate of all proposed time slots, and then evaluate the attendance of each time slot. The system should assign the meeting to the time slot with the maximum attendance if using the maximum attendance approach.

In the next section, we compare the above algorithms in detail by using computer simulations to find the earliest common time slots among the proposed time slots.

6.2 Analytical Results and Discussions

To perform the simulations, the following parameters are used. Ten users are invited to attend the meeting. The attendance rate should be over 80%. Five time slots are proposed. The availability of each invitee is 0.8 and the reply probability is 0.8.

The simulation program was written in Java. For each set of data, the simulation was repeated 10,000 times to obtain the average result. The average scheduling cost is calculated by counting the number of checkings. When the system checks the availability of the invitee at a particular time slot, the scheduling cost is increased by 1. For example, the system performs thirteen checks if using the random matching algorithm in the previous example. The corresponding scheduling cost is 13. When the system starts a new cycle to determine whether the time slot is a common time slot or not, the number of matching cycles increases by 1. For example, the system checks three time slots and then finds a common time slot; the number of cycles is three. The simulations were conducted by changing the availability, attendance rate, and reply rate of the invitees.

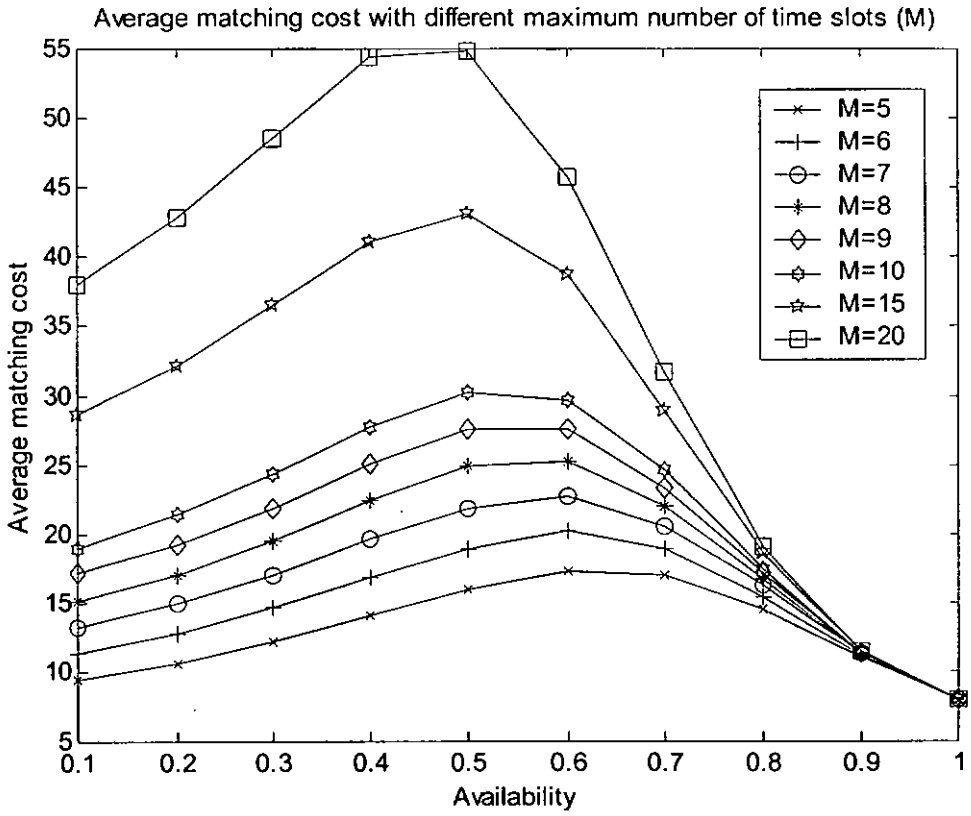

Figure 78. Average scheduling cost with different maximum number of proposed time slots

Figures 78, 79 and 80 show the average scheduling cost, number of cycles and average success rate when the availability changes from 0.1 to 1.0. Figure 71 shows that the average scheduling cost rises when the availability decreases from 1.0 to 0.6 and, if the system proposes five to nine time slots, it drops slightly when the availability decreases from 0.6 to 0.1. Moreover, if the system proposes ten, fifteen and twenty time slots, the average scheduling costs increases until the availability decreases from 1.0 to 0.5. Furthermore, the average scheduling cost is high when the number of proposed time slots is high. This is because the system needs to check more time slots if more time slots are proposed, as, for example, if the system proposes five time slots only. In the worse case, the common time slot is the 5^{th} time slot (i.e., the system checks all

time slots). As a result, the scheduling cost increases as more time slots are proposed.

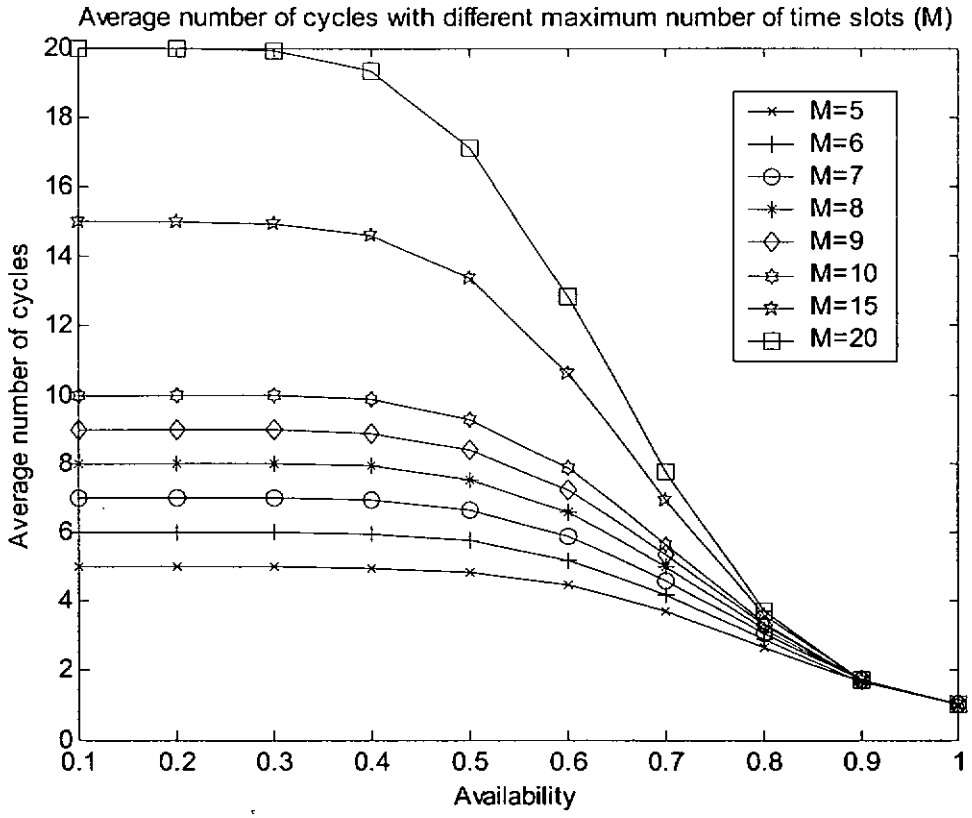

Figure 79. Average number of cycles with different maximum number of proposed time slots

As shown in Figure 78, the average matching cost increases slightly when the availability drops from 1.0 to 0.5. When the availability decreases, the system needs to check more time slots among the invitees, in order to find a common time slot. On the other hand, when the availability is less than 0.5, the average scheduling cost decreases. The system performs fewer checks because it only needs to check few invitees at the each time slot due to the high unavailability of the invitees. As a result, the scheduling cost decreases when the availability is less than 0.5.

Figure 79 shows that the average number of cycles is required when the availability changes from 0.1 to 1.0. The average number of cycles increases when the availability decreases from 1.0 to 0.5. The average number of cycles increases when the availability decreases, because the system needs to check more time slots to find the common time slot if the availabilities of the invitees are small. As shown in Figure 79, the system needs to check all the proposed time slots when the availability is less than 0.4.

Figure 80 shows the average success rate for matching a time slot when the availability of the invitees changes from 1.0 to 0.1. Obviously, the success rate decreases sharply if the availability decreases from 0.9 to 0.3. The success rate tends to be zero when the availability is less than 0.4. Furthermore, when the number of proposed time slots increases, the success rate increases too when the availability is between 0.4 and 0.8.

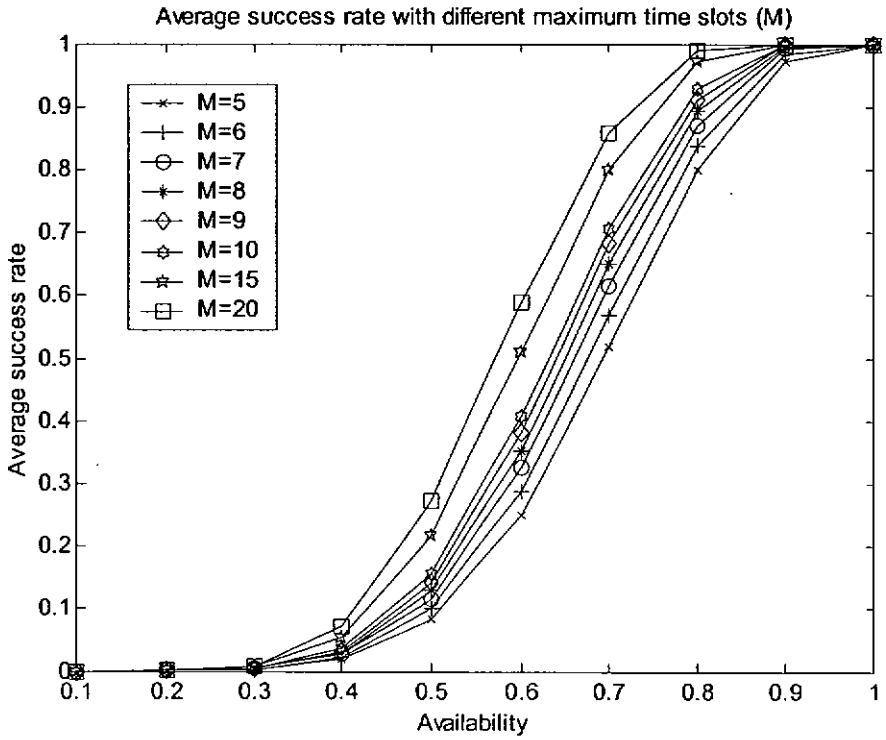

Figure 80. Average success rate with different maximum number of proposed time slots

Based on the result, the scheduling cost increases when the number of the proposed time slots increases, and the success rate increases too. As a result, the system proposes more time slots will cause the increasing of the scheduling cost, but the system finds a common time slot for a meeting easier.

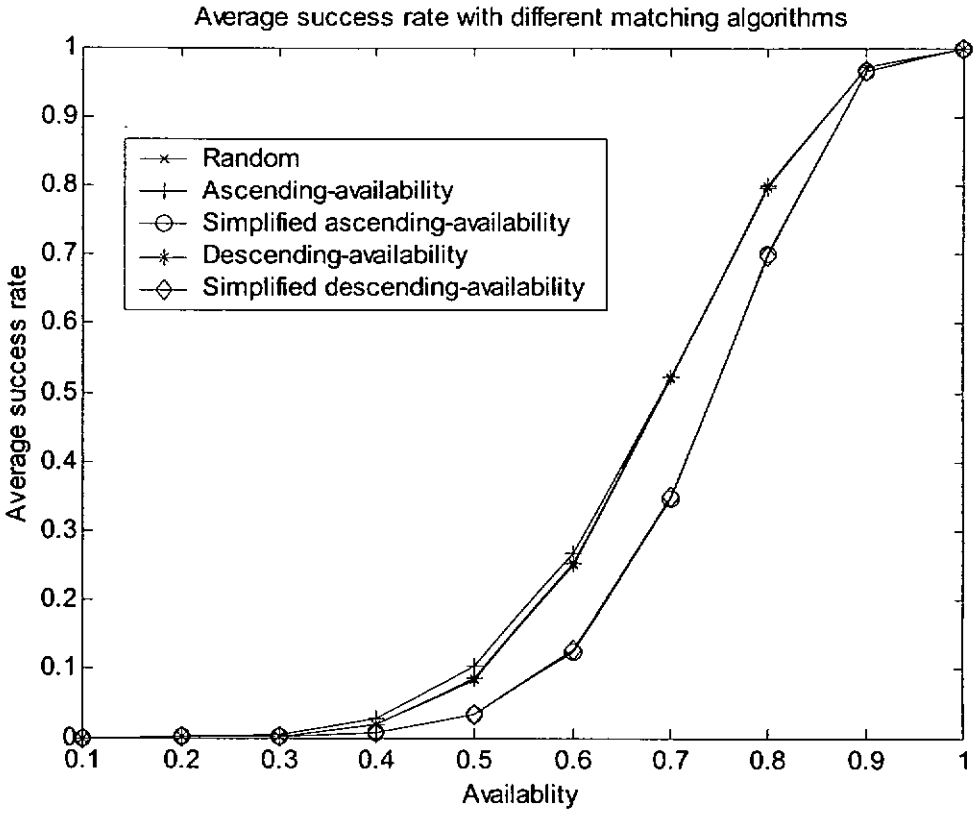

Figure 81. Average success rate with different matching algorithms

Before we investigate how the matching cost affected by the availability and by the matching method, next we analyze on the relationship between the average success rate and the availability of the invitees. As shown in Figure 81, the average success rate of the random matching algorithm, ascending-availability matching algorithm and descending-availability matching algorithm are higher than that of the simplified ascending-availability matching algorithm and the simplified descending-availability matching algorithm when the availability is

between 0.4 and 0.8. In addition, the average success rates of the random matching algorithm, ascending-availability matching algorithm and descending-availability matching algorithm are close to each other. It can be seen that the success rate of the simplified ascending-availability matching algorithm and the simplified descending-availability matching algorithm are almost the same when the availability is between 0.1 and 1.0.

Figure 82 shows that the average number of cycles needed for the random matching algorithm, ascending-availability matching algorithm and descending-availability matching algorithm are less than those of the simplified ascending-availability matching algorithm and simplified descending-availability matching algorithm when the availability is between 0.5 and 0.9. According to Figure 82, the difference between two sets of data is less than 1 cycle. The greatest difference occurs when the availability is equal to 0.7 and 0.8.

The average number of cycles of the simplified ascending-availability matching algorithm and the simplified descending-availability matching algorithm are higher than that of the other three matching algorithm methods, because both of the simplified matching algorithms start to check a new time slot when any one of the invitees gives a negative reply for the current time slot, i.e., he/she is not available at the current time slot.

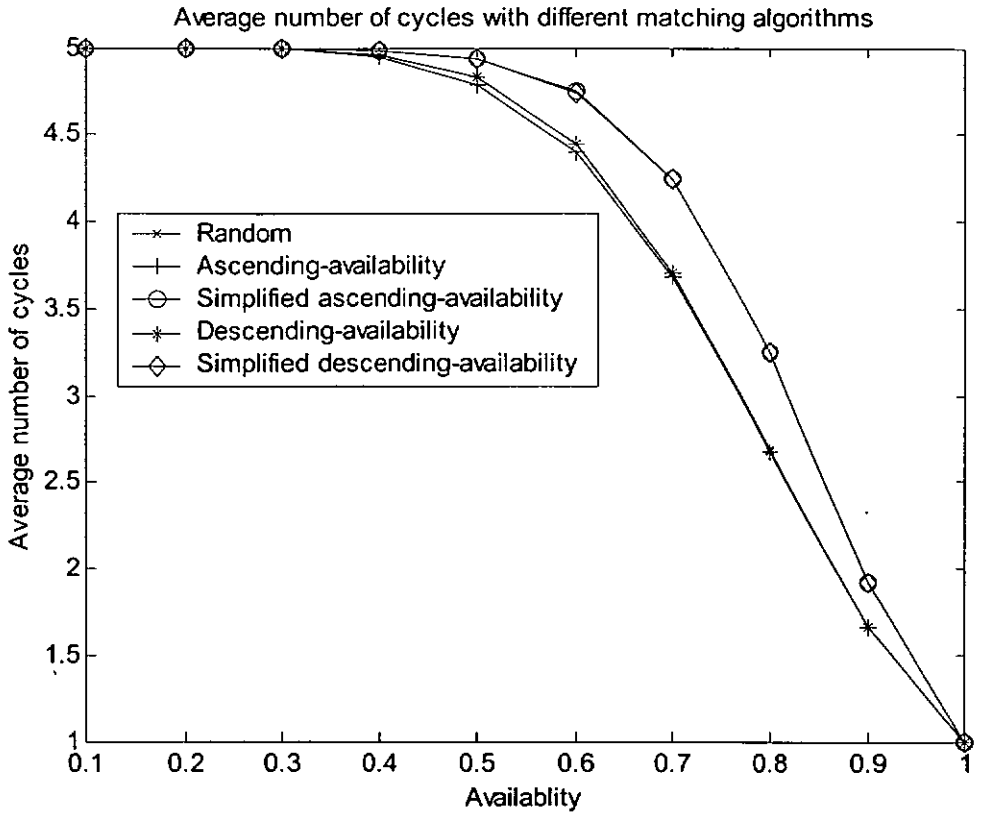

Figure 82. Average number of cycles with different matching algorithms

Figure 83 shows the relationship between the average scheduling cost and the matching algorithm when the availability changes from 0.1 to 1.0. It can be seen that the descending-availability matching algorithm has the highest average matching cost when the availability is less than 0.8. However, when the availability is between 0.8 and 1.0, the simplified descending-availability matching algorithm has the highest average scheduling cost, while the simplified ascending-availability matching algorithm has the lowest average matching cost.

As mentioned in the Figures 81 and 82, the success rate and the average number of the matching cycles of the random matching algorithm, ascending-availability matching algorithm and descending-availability matching algorithm are closed to each other when the availability increases from 0.1 to 1.0. The descending-availability matching algorithm has the highest average matching cost when

compared with the other two methods, while the ascending-availability matching algorithm gives the lowest matching cost. As shown in Figure 83, the matching cost of the descending-availability matching algorithm is two times of the ascending-availability matching algorithm when the availability is equal to 0.6. In this case, it is better to use the ascending-availability matching algorithm to determine the common time slot for a meeting.

Comparing the average matching cost of the simplified ascending-availability matching algorithm and that of the simplified descending-availability matching algorithm, the simplified ascending-availability matching algorithm has a lower matching cost. The matching cost difference between them is in fact very large.

Both the ascending-availability matching algorithm and the simplified ascending-availability matching algorithm can provide a better matching cost on compared with the other methods. In addition, the average success rate of the ascending-availability matching algorithm is higher than that of the simplified ascending-availability matching algorithm, but its average matching cost is higher.

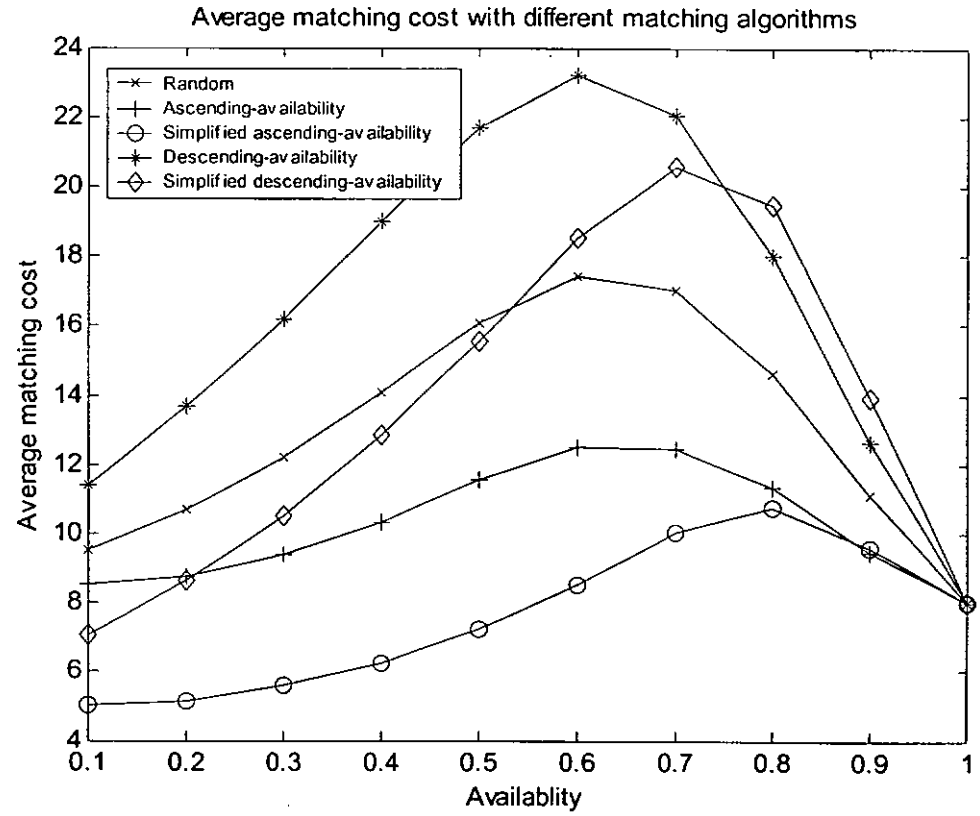

Figure 83. Average matching cost with different matching algorithms

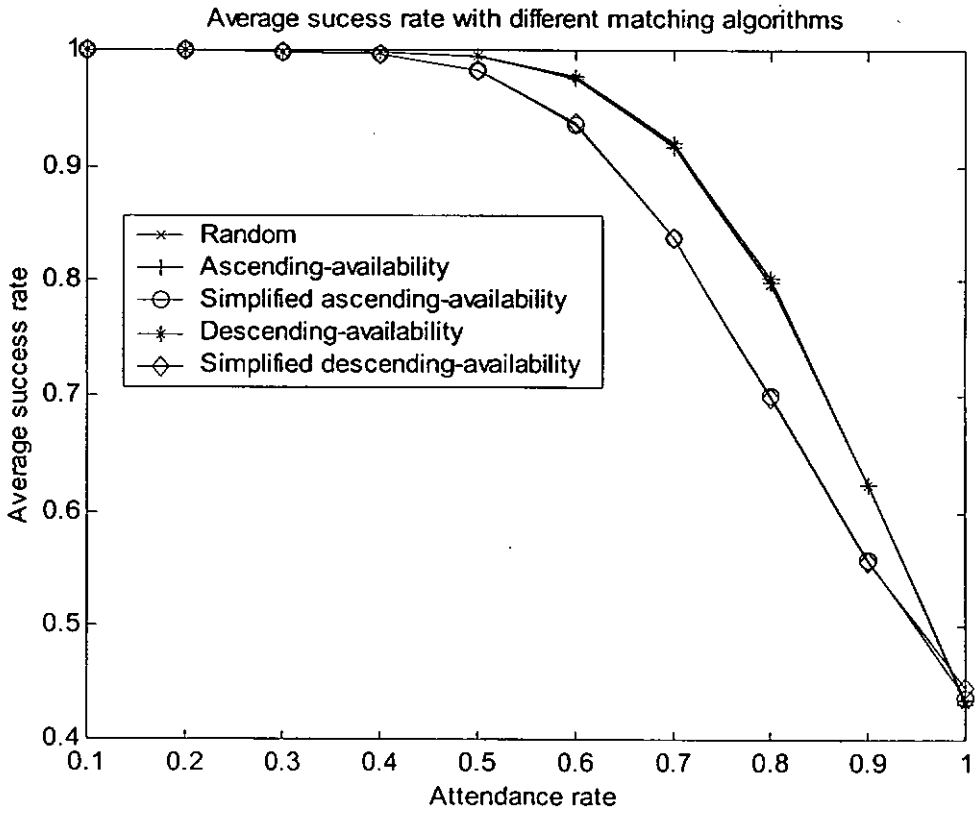

Figure 84. Average success rate of different matching algorithms

Figure 84 shows that the average success rate increases sharply when the attendance rate decreases from 1.0 to 0.6. When the attendance rate is less than 0.6, the average success rate tends to be 1.0. According to Figure 84, the average success rates of the random matching algorithm, ascending-availability matching algorithm and descending-availability matching algorithm are higher than that of the simplified ascending-availability matching algorithm and the simplified descending-availability matching algorithm when the attendance rate is between 0.5 and 0.9. On the other hand, the average success rates of all matching algorithms merge together when the attendance rate is equal to 1.0 and less than 0.5.

Based on Figure 84, the five matching algorithms can be classified into two groups. The random matching algorithm, ascending-availability matching algorithm and descending-availability matching algorithm can be classified into same group, because their average success rates are close to each other. On the other hand, the simplified ascending-availability matching algorithm and the simplified descending-availability matching algorithm give a similar performance.

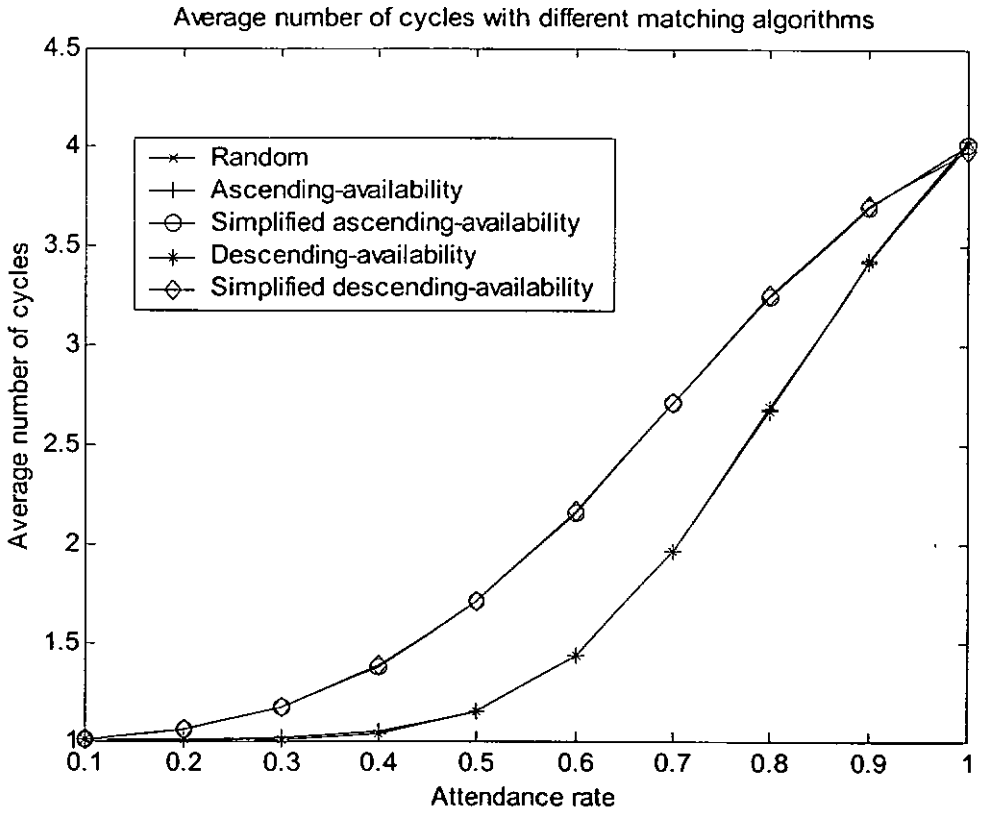

Figure 85. Average number of cycles of different matching algorithms

Figure 85 shows the effect of the attendance rate on the average number of cycles. The result is similar to that presented in Figure 84. The 5 matching methods can be classified into two groups based on the results. i.e., the random matching algorithm, ascending-availability matching algorithm and the descending-availably matching algorithm belong to one group, while the simplified ascending-availability matching algorithm and the simplified descending-availability matching algorithm form another group. As shown in Figure 85, when the attendance rate increases, the average number of cycles increases too. This is because when the attendance rate is high, more attendees have to attend the meeting, so the system needs to check more time slots in order to find a common time slot among the attendees. Furthermore, the average number of cycle for the random matching algorithm, ascending-availability

matching algorithm and descending-availability matching algorithm is lower than that of the simplified ascending-availability matching algorithm and the simplified descending-availability matching algorithm.

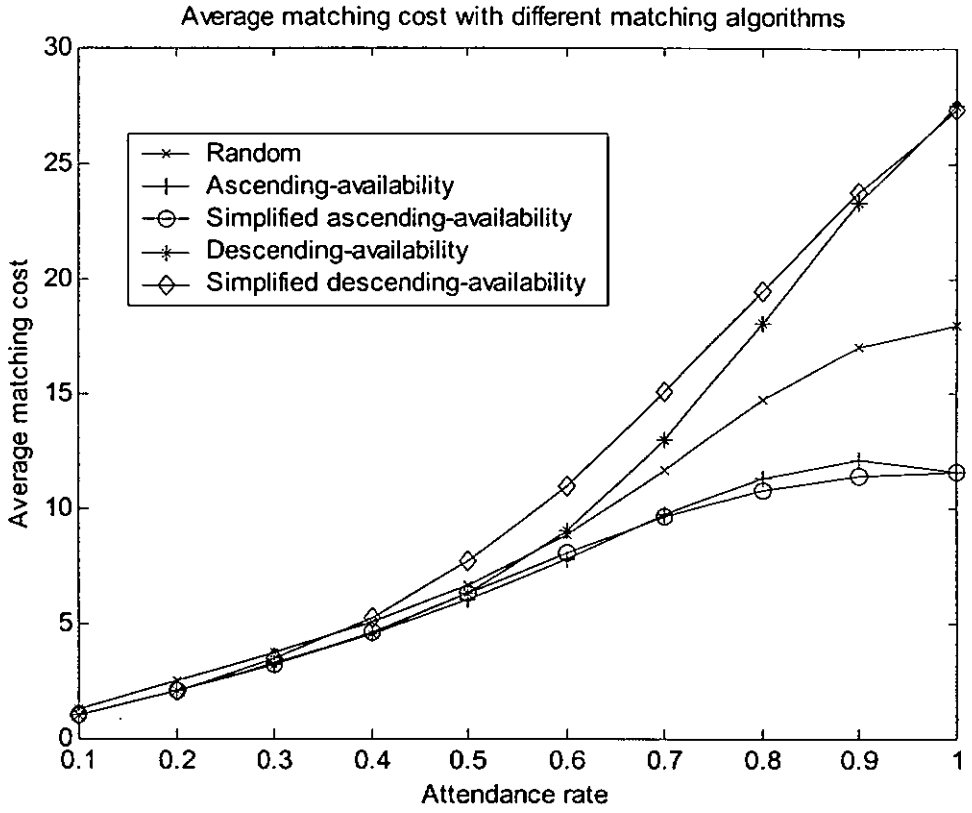

Figure 86. Average number of cycles of different matching algorithms

It can be seen that the average matching cost increases when the attendance rate increases. This is because when the attendance rate is high, more invitees should attend the meeting. As a result, the system has to check more time slots (as shown in Figure 85) thus resulting in a high matching cost. As shown in Figure 86, the meeting matching cost of the simplified descending-availability matching algorithm has the highest average matching cost, while the simplified ascending-availability matching algorithm has the lowest average matching cost.

Furthermore, the average matching costs of all the meeting scheduling algorithms are close to each other when the attendance rate is less than 0.5. Concerning the random matching algorithm, ascending-availability matching algorithm and descending-availability matching algorithm, we found that the ascending-availability matching algorithm can find a common time slot with the minimum matching cost when the attendance rate is between 0.1 and 1.0. Moreover, the descending-availability matching algorithm gives the highest matching cost.

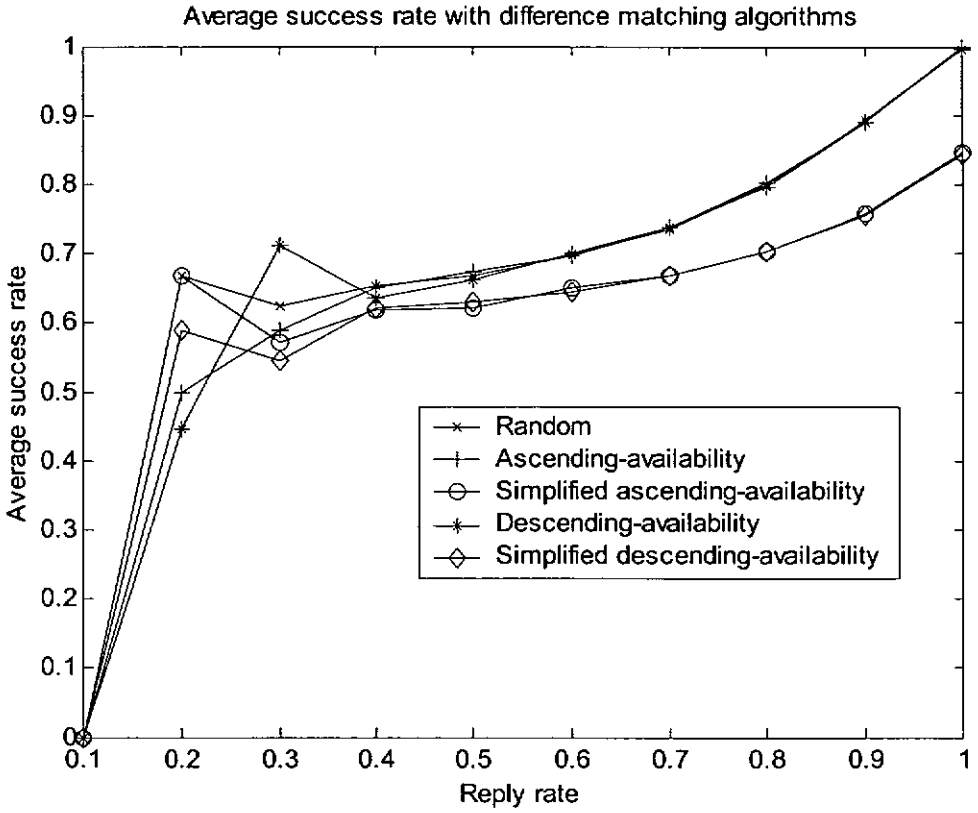

Figure 87. Average success rate of different matching algorithms

Note that the system starts to check the common time slot only when at least 8 invitees reply to the system. If there is insufficient number of replies, the meeting will be scheduled unsuccessfully and the matching cost is equal to zero. As shown in Figure 87 when the reply rate of the invitees decreases, the success

rate also decreases slightly, except when the reply rate is equal to 0.1. When the reply rate is equal to 0.1, the success rate drops to zero, because there are insufficient number of replies. As a result, the system cannot find a common time slot and the success rate is equal to zero.

As shown in Figure 87, the success rate of the random matching algorithm, ascending-availability matching algorithm and the descending-availability matching algorithm are close to each other. Their success rates are higher than those of the simplified ascending-availability matching algorithm and the simplified descending-availability matching algorithm. In addition, the curves of the two simplified matching algorithms merge when the reply rate is between 0.1 and 1.0.

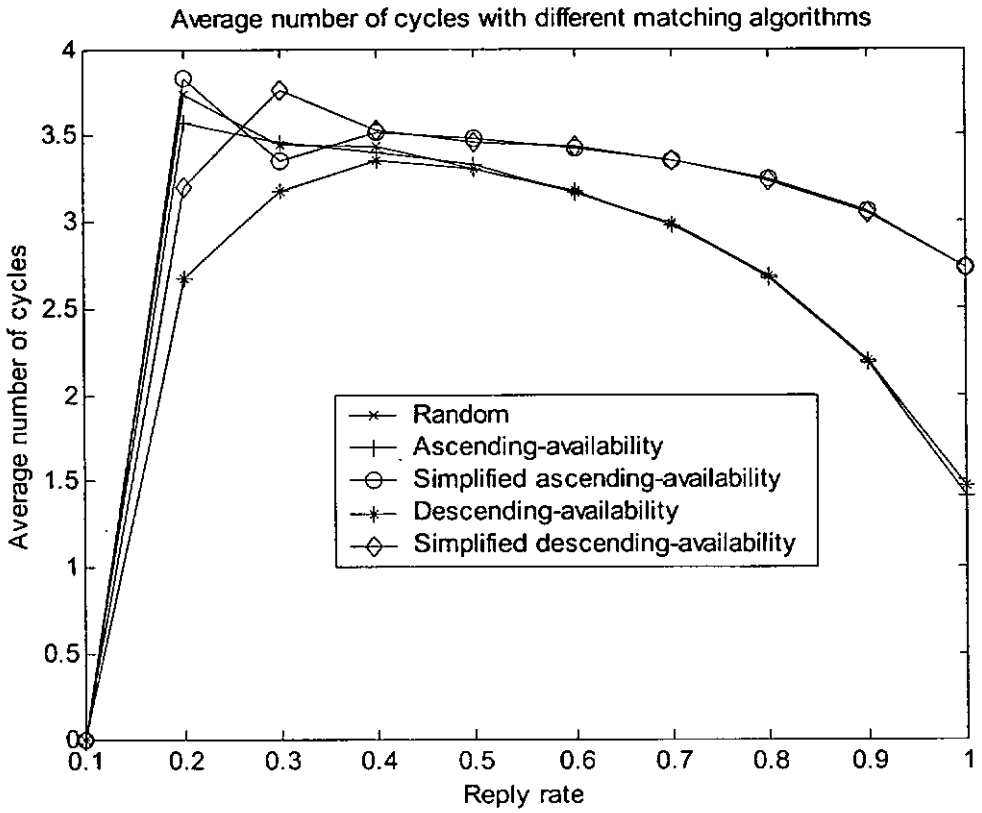

Figure 88. Average number of cycles of different matching algorithms

Figure 88 shows that when the reply rate decreases from 1.0 to 0.3, the average number of cycles increases from 1.5 to 3. When fewer invitees reply to the system, the system should check more time slots in order to find the common time slot among the repliers. The average number of cycles drops to zero when the reply rate is equal to 0.1, because the system cannot receive sufficient number of replies from the invitees.

As shown in Figure 88, the average number of cycles of the random matching algorithm, ascending-availability matching algorithm and the descending-availability matching algorithm merge when the reply rate decreases from 1.0 to 0.3. In addition, the average number of cycles of the simplified ascending-availability matching algorithm and the simplified descending-availability matching algorithm are close to each other too. In general, the random matching algorithm, ascending-availability matching algorithm and the descending-availability matching algorithm take fewer cycles for finding a common time slot.

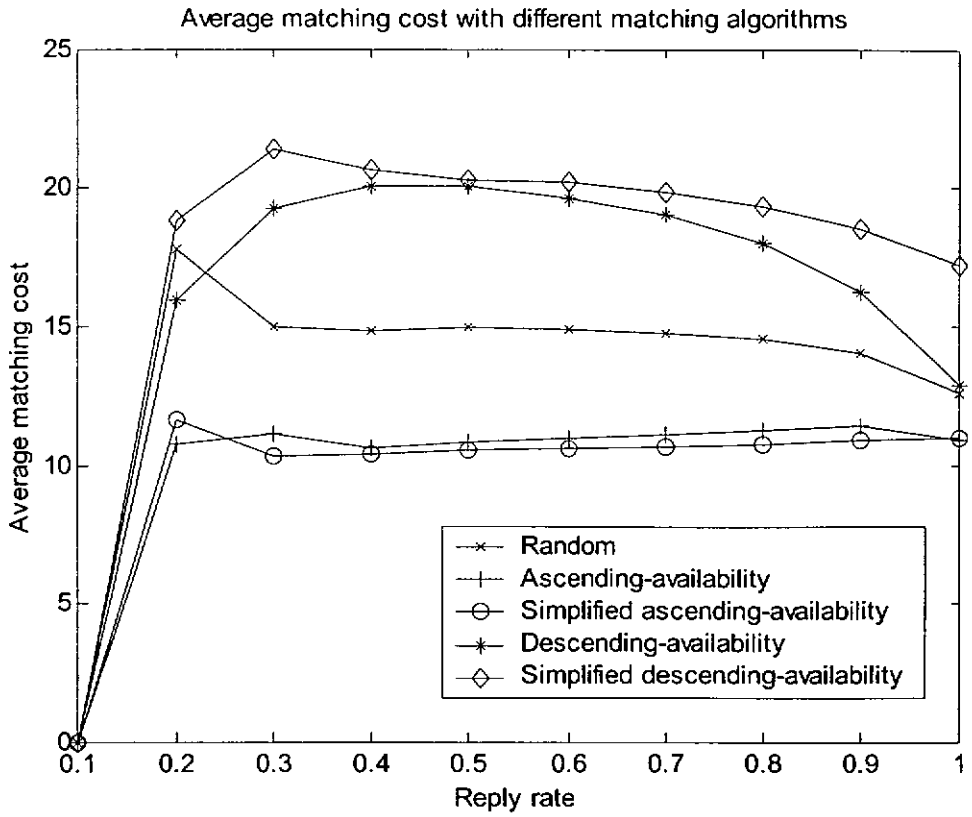

Figure 89. Average matching cost of different matching algorithms

Figure 89 shows that the simplified ascending matching algorithm has the lowest average matching cost, while the simplified descending matching algorithm has the highest average matching cost among the 5 matching algorithms. The simulation result is similar to the results for changing the availability of the invitees or the attendance rate. As shown in Figure 89, the ascending-availability matching algorithm provides the minimum average matching cost, while the descending-availability matching algorithm has the highest average matching cost.

Furthermore, when the reply rate decreases from 1.0 to 0.3, the average matching cost of the matching algorithms remains almost constant, except that the matching cost of the descending-availability matching algorithm and the simplified descending-availability matching algorithm increases slightly. The

reply rate of the invitees does not affect the average matching cost. This is because the system only finds the common time slot when a sufficient number of replies is received. As the availability of the invitee does not affect the reply rate, the matching cost is insensitive in the reply rate of the invitees.

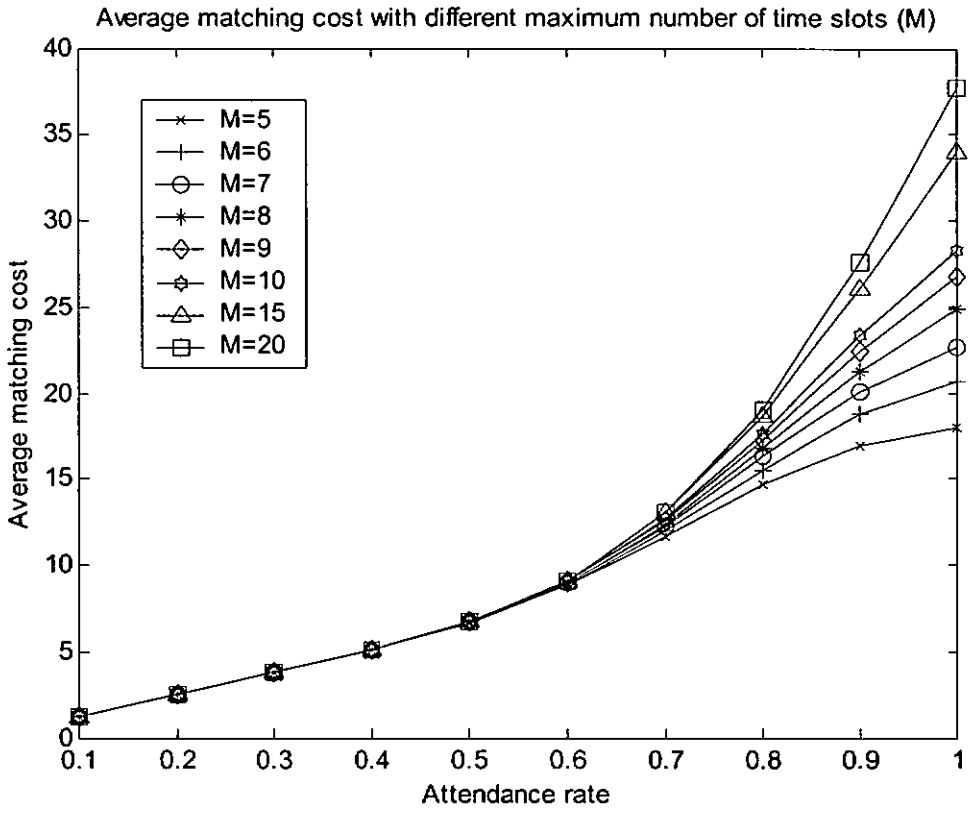

Figure 90. Average matching cost of different maximum number of proposed time slots

Figure 90 shows that the average matching cost increases when the attendance rate increases from 0.6 to 1.0. Furthermore, when the number of proposed time slots increases, the average matching cost increases too. This is because when the attendance rate is high, more attendees are required to attend the meeting, and the system needs to check more time slots, in order to find the common time slot. Also, when the system proposes more time slots, it tries to check as many time slots as possible to find the common one among the invitees. As a result, the matching cost increases when the number of proposed time slots increases.

Also, the number of proposed time slots affects the success rate of scheduling a meeting and the number of cycles needed. The effect on the average number of cycles and the average success rate are shows in Figures 90 and 91.

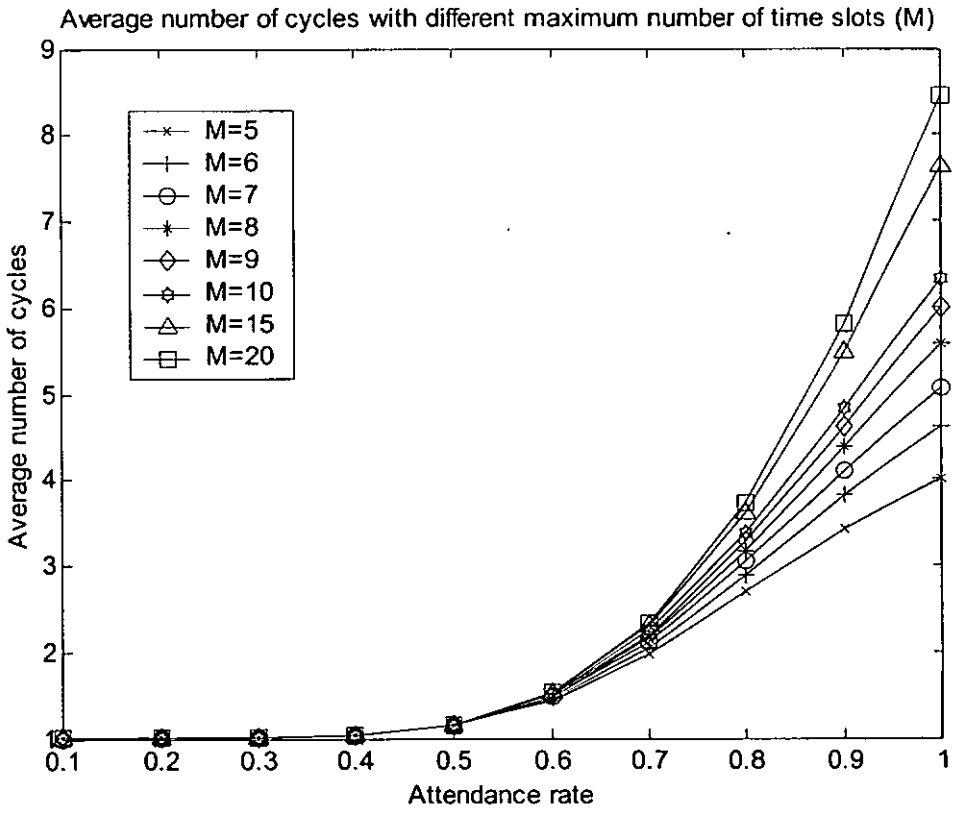

Figure 91. Average number of cycles of different maximum number of proposed time slots

Figure 91 shows that as the attendance rate increases, the average number of cycles required also increases. This is because the system needs to check more time slots, in order to find the common time slot when the minimum required number of attendees increases. Again, if the attendance rate is greater than 0.7 and the system proposes more time slots in each iteration, the system will prefer to check more time slots, in order to find the common time slot for a meeting. As a result, if the value of M increases, the average number of matching cycles also

increases. On the other hand, the success rate also increases when the system proposes more time slots. The result is shown in figure 92.

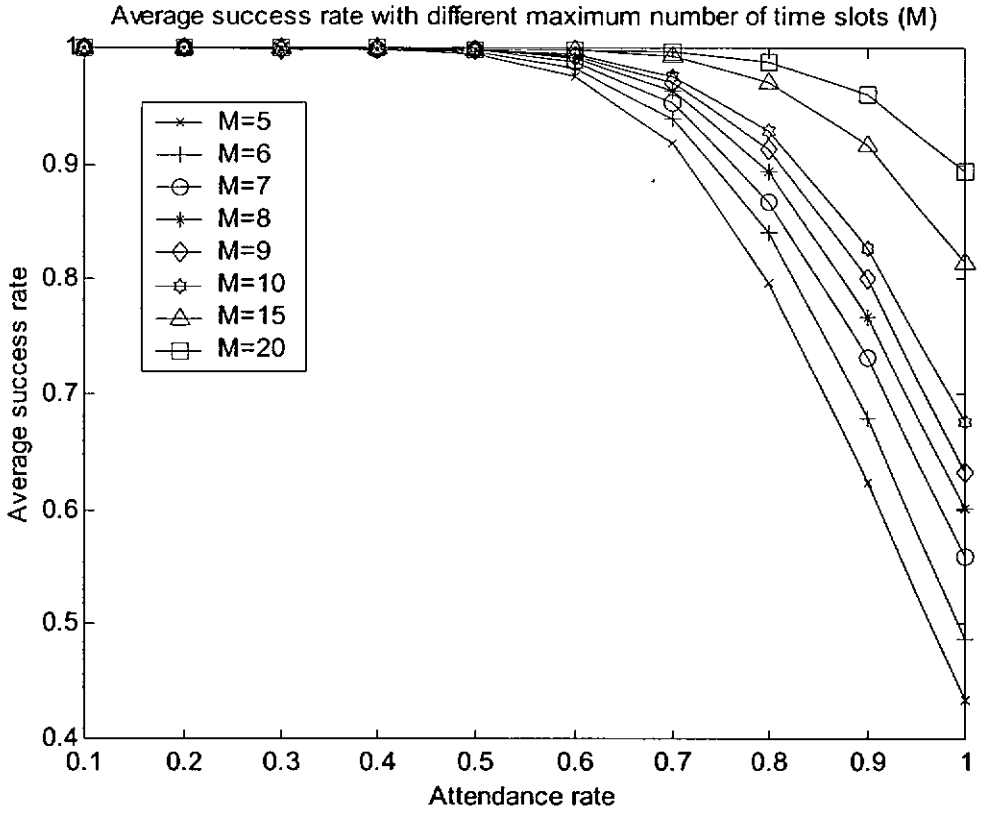

Figure 92. Average success rate of different maximum number of proposed time slots

When the attendance rate decreases from 1.0 to 0.5, the average success rate increases (as shown in Figure 92). The success rate is also affected by the number of proposed time slots. When more time slots are proposed to the invitees, the system more easily finds a common time slot within the proposed time slots.

6.3 Summary

In this chapter, we have proposed five different matching algorithms for finding a common time slot for a meeting. They can be classified based on whether sorting is required. The random matching algorithm is a non-sorting based matching algorithm. The other four matching algorithms sort the order of the repliers based on their availabilities. The algorithms are ascending-availability matching algorithm, descending-availability matching algorithm, simplified ascending-availability matching algorithm and simplified descending-availability matching algorithm.

By analyzing these algorithms using computer simulations, we found that the average number of cycles and the average success rate of the random matching algorithm, the ascending-availability matching algorithm and the descending-availability matching algorithm are close to each other, when the availability of the invitees, attendance rate and reply rate are changed. In addition, the average number of cycles and the average success rate of the simplified ascending-availability matching algorithm and simplified descending-availability matching algorithm give similar performance too. Moreover, the ascending-availability matching algorithm gives the lowest matching cost when compared with the random matching algorithm and the descending-availability matching algorithm. And the descending-availability matching algorithm is the most highest expensive, irrespective of the availability of the invitees, the attendance rate and the reply rate.

CHAPTER 7
CONCLUSION

Inspired by existing virtual secretary systems, we have proposed an electronic secretary system called ASSIST, which stands for Automatic Secretary System using Internet and Smart-agent Technologies. As its name implies, ASSIST provides simple secretarial tasks such as meeting scheduling, document management, email filtering etc. Like other electronic secretaries, the focus of this thesis is on meeting scheduling.

Most of the existing meeting scheduling systems propose an arbitrary number of time slots to the invitees for scheduling purposes. In order to provide a more effective mechanism, we have proposed two methods to find the optimal number of time slots for scheduling a meeting. In the first model as presented in Chapter 4, we have determined the fixed number of time slots per scheduling cycle in order to minimize scheduling cost. It is found that on average two to three time slots should be suggested. Furthermore, one to two iteration is required to schedule a meeting successfully. In addition, we found that as the length of the meeting, overhead cost or attendance rate of the meeting increases, the optimal number of proposed time slots increases too. However, if the availability is less than 0.6, the optimal number of time slots is insensitive to change of the aforementioned parameters.

As the model presented in Chapter 4 does not consider the scheduling deadline, we have formulated a Markov decision model in Chapter 5 to address the issue. We have used the backward induction method to find the optimal number of proposed time slots at each decision stage when scheduling a meeting. By doing so, the number of time slots can be varied dynamically when approaching the deadline. To find a common time slot, we have proposed five matching algorithms for finding a common time slot. They are: random matching algorithm, ascending-availability matching algorithm, descending-availability matching algorithm, simplified ascending-availability matching algorithm and simplified descending-availability matching algorithm. We found that the ascending-availability matching algorithm gives the lowest matching cost and the highest successful rate when compared with the other matching algorithms.

Using the mathematical model and a Markov decision model would allow a meeting to be scheduled at a minimal scheduling cost. The mathematical model suggests proposing 2 to 3 time slots in each iteration, A common time slot can be found within 2 iterations. The findings match with our daily experience, because we always propose 2 to 3 time slots when scheduling a meeting. The Markov decision model suggests proposing different numbers of time slots at different decision points. As a result, we can develop a meeting scheduling system using the mathematical model or Markov decision model, in order to schedule a meeting with the minimal scheduling cost. In addition, we can implement the ascending-availability matching algorithm into the scheduling system, so the system uses the best strategy to find the common time slot after proposing the time slots when scheduling a meeting. As a result, the meeting

scheduling system schedules a meeting in an effective and efficiency way by minimizing the scheduling cost and matching time slot cost.

REFERENCES

[1] A. Ashir , G. Mansfield and N. Shiratori, A Meeting Scheduling System for Global Events on the Internet, http://www.isoc.org/inet98/proceedings/1b/1b_1.htm.

[2] A. Ashir, K. H. Joo, T. Kinoshita and N. Shiratori, "Multi-agent based decision mechanism for distributed meeting scheduling system", *Proc. International Conference on Parallel and Distributed Systems*, pp. 275 - 280, December 1997.

[3] L. Adams, L. Toomey and E. Churchill, "Distributed research teams: meeting asynchronously in virtual space", *Proc. 32nd Annual Hawaii International Conference on System Sciences (HICSS 32)*, vol 2, pp. 10, 5 – 8 January 1999.

[4] J. G. Bellika, G. Hartvigsen and R. A. Widding, "Using user models in software agents: the virtual secretary", *Proc. 1998 International Conference on Multi Agent System*, pp. 391-392, July 1998.

[5] C. C. Bian, W. Cao and G. Hartvigsen, "ViSe2-an agent-based expert consulting system with efficient cooperation", *Proc. IEEE International Conference on Intelligent Engineering Systems (INES 97)*, pp. 191 – 196, 15 – 17 September 1997.

[6] D. Boyer, M. Cortes, M. Vernick, S. Wilbur, A. Khan and G. Balfour, "Virtual social clubs: meeting places for the Internet community", *Proc. IEEE International Conference on Multimedia Computing and Systems*, vol. 2, pp. 297 – 301, 7 – 11 June 1999.

[7] S. Bocionek, "Software secretaries: learning and negotiating personal assistants for the daily office work", *Proc. 1994 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 1, pp. 7-12, October 1994.

[8] H. H. Bui, S. Venkatesh and D. Kieronska, "A multi-agent incremental negotiation scheme for meetings scheduling", *Proc. 3rd Australian and New Zealand Conference on Intelligent Information Systems 1995 (ANZIIS-95)*, pp. 175 – 180, 27 November 1995.

[9] D. A. Chappell, D. R. Vogel and E. E. Roberts, "The MIRROR project: a virtual meeting place", *Proc. 25th Hawaii International Conference on System Sciences*, vol. 4, pp. 23 – 33, 7 – 10 January 1992.

[10] CONCORDIA, http://www.merl.com/HSL/Projects/Concordia/.

[11] W. Cao, C. G. Bian and G. Hartvigsen, "Cooperator-Base + Task-Base for Agent Modeling: the Virtual Secretary Approach", *Proc. 13th National Conference on Artificial Intelligence, Agent Modeling Workshop (AAAI'96)*, pp. 105-111, 1996.

[12] W. Cao, C. G. Bian and G. Hartvigsen, "ViSe2 - An Application of Intelligent Agents Approaching Efficient Cooperation in a Multi-Agent System", *Proc. 6th Scandinavian Conference on Artificial Intelligence* (SCAI'97), Helsinki, Finland, pp. 176 – 186, 1997.

[13] P. H. Dawkins, "The impact of expert systems in office automation", *Proc. IEEE Colloquium on Emerging Office Technologies*, pp. 4/1 – 4/3, 20 April 1988.

[14] E.C. Freuder, M. Minca and R.J. Wallace, "Privacy/Efficiency Tradeoffs in Distributed Meeting Scheduling by Constraint-Based Agents", *Proc. IJCAI DCR Workshop*, pp. 63- 72, 2001.

[15] M. V. Genuchten , W. Cornelissen and C. V. Dijk, "Supporting inspections with an electronic meeting system", *Proc. 30th Hawaii International Conference on System Sciences*, vol. 2, pp. 405 – 411, 7 – 10 January 1997.

[16] A. B. Hassine, T. Ito and T. B. Ho, "A new distributed approach to solve meeting scheduling problems", *Proc. IEEE/WIC International Conference on Intelligent Agent Technology 2003 (IAT 2003)*, pp. 588 – 591, 13 – 17 October 2003.

[17] G. Hartvigsen, S. Johansen, A. Helme, R. A. Widding, G. Bellika and W. Cao, "The Virtual Secretary Architecture for Secure Software Agents", *Proc. 1st International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 96)*, 22-24 April 1996. (Poster)

[18] G. Hartvigsen, V. Farsi and B. Vinter, "The Virtual Secretary Project: Some Parts of the Project's Theoretical Background", Virtual Secretary Working Report 95-02. Department of Computer Science, University of Tromso

[19] G. Hartvigsen, S. Johansen, R.A. Widding, G. Bellika and W. Cao, "The Virtual Secretary", Virtual Secretary Working Report 95-03. Department of Computer Science, University of Tromso

[20] T. Haynes, S. Sen, N. Arora and R. Nadella, "An Automated Meeting Scheduling System That Utilizes User Preferences", *Proc. 1st*

International Conference on Autonomous Agents, pp. 308-315, Marina del Rey, CA, USA, February 1997.

[21] K. Higa, B. Shin and V. Sivakumar, "Meeting scheduling: an experimental investigation", *Proc. IEEE International Conference on Systems, Man, and Cybernetics 1996*, vol. 3, pp. 2023 – 2028, 14 – 17 October 1996.

[22] I. S. K. Ho and H. C. B. Chan, "A Markov decision-based meeting scheduling mechanism for automatic secretary system using internet and smart-agent technologies (ASSIST)", *Proc. IEEE International Conference on Systems, Man and Cybernetics 2003*, vol. 5, pp. 4552 – 4559, 5 – 8 October 2003.

[23] I. S. K. Ho and H. C. B. Chan, "ASSIST: Automatic Secretary System using Internet and Smart-agent Technologies for scheduling meetings", *5th International Conference on Electronic Commerce*, 30 September – 3 October 2003 (poster)

[24] T. Ide, A. Hakata and W. M. Kim, "An intelligent network service prototype using knowledge processing", *Proc. 3rd International Conference on Tools for Artificial Intelligence 1991 (TAI '91)*, pp. 445 – 448, 10 – 13 November 1991.

[25] N. R. Jennings and A. J. Jackson, "Agent-based meeting scheduling: a design and implementation", *Electronics Letters*, 31(5), pp. 350-352, March 1995.

[26] W. S. Jeong, J. S. Yun and G. S. Jo, "Cooperation in multi-agent system for meeting scheduling", *Proc. IEEE Region 10 Conference (TENCON 99)*, vol. 2, pp. 832 – 835, December 1999.

[27] J. Klein, "Advantages of office automation application in concurrent engineering environment", *Proc. Portland International Conference on Management and Technology (PICMET 97)*, pp. 887, 27 – 31 July 1997.

[28] M. W. Kim, J. Maeda, R. Yatsuboshi, S. Hattori and W. S. Meeks, "An advanced service application for private intelligent secretary service", *Proc. International Switching Symposium*, vol. 3, pp. 33 – 36, 27 May – 1 June 1990.

[29] Z. Koono, P. Ma, H. Chen, B. H. Far, M. M. El-Khouly, "Human knowledge to be applied to communications services", *Proc. International Conference on Communication Technology 1998 (ICCT 98)*, vol. 2, pp. 5, 22 – 24 October 1998.

[30] D.B. Lange and M. Oshima, *Programming and Deploying Java Mobile Agents with Aglets*, Addison-Wesley, 1998

[31] M. Masoodian and S. Luz, "Heterogeneous client-server architecture for a virtual meeting environment", *Proc. 8th Euromicro Workshop on Parallel and Distributed Processing 2000*, pp. 67 – 74, 19 – 21 January 2000.

[32] C. H. Park and J. Y. Kwak, "A secretary agent system based on HTTP client-server protocol", *Proc. IEEE Region 10 Conference (TENCON 99)*, vol. 2, pp. 1087 -1090, December 1999.

[33] Martin L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley & Sons, Inc., New York, NY, 1994.

[34] S. Schmeier, A. Schupeta, "PASHA – Personal Assistant for Scheduling Appointments", *Proc. 1st Conference on Practical Application of Multi Agent Systems*, London 1996.

[35] S. Sen, "Developing an automated distributed meeting scheduler", *IEEE Expert*, vol. 12, no. 4, pp. 41 - 45, July/August 1997.

[36] S. Sen and E. H. Durfee, "A formal study of distributed meeting scheduling", *Proc. Group Decision and Negotiation*, vol. 7, pp. 265-289, 1998.

[37] S. Sen and E. H. Durfee, "On the design of an adaptive meeting scheduler", *Proc. 10th IEEE Conference on Artificial Intelligence for Applications*, pp. 40 - 46, March 1994.

[38] S. Sen and E. H. Durfee, "Using temporal abstractions and cancellations for efficiency in automated meeting scheduling", *Proc. International Conference on Intelligent and Cooperative Information Systems*, pp. 163 - 172, May 1993.

[39] C. H. Yang, S. L. Yen, H. D. Liu, K. Liu, B. S. Jeng, K. Y. Chan, M. S. Chang, Y. L. Cheng, J. L. Liang and D. M. Shien, "Secure official document mail systems for office automation", *Proc. International Carnahan Conference on Security Technology*, pp. 161 – 164, 15 - 17 October 1997.

SYMBOLS

Symbols used in the scheduling model in chapter 4 and chapter 5

m	The number of time slots proposed in each cycle
i	The i^{th} iteration/cycle
Q	Probability that a meeting can be scheduled at the i^{th} iteration/cycle
f	Cost of filling in a time slot
o	Overhead cost of each scheduling cycle
p	Probability that an invitee is free at a time slot
d	Required number of time slot(s) for a meeting
q	Probability that an invitee can attend the meeting
G	Number of groups
j	the j^{th} meeting group
A_j	Minimum attendance rate of group M_j
I_j	Number of people in group M_j
N_j	Minimum number of attendees in group M_j
M	The maximum number of time slots proposed in each cycle
l	Cost of locking a proposed time slot
C	Cost of failing to schedule a meeting
r	Probability of an invitee replying to the system (response rate)
R	Probability that the sufficient number of invitees reply to the system

D	Scheduling deadline
n	Number of outstanding responses
W	Waiting state
E	End state
F	Failure state
$W_{m,n}$	Waiting state with m number of proposed time slots and n number of outstanding responses
$T(\Omega_{t+1}\|\Omega_t,\Theta_t)$	Transition probability that the system state change from Ω_t at t to Ω_{t+1} at t+1 and the action taken at t is Θ_t
Ω_t	System state at t
Θ_t	Action taken at t
$Z_{t+1}^*(F)$	Minimal average scheduling cost from t+1 to D when the system state is F
$\delta_t(F,m)$	Immediate cost at t