Available online at www.sciencedirect.com
SCIENCE DIRECT®
Engineering Applications of Artificial Intelligence 16 (2003) 727–743
Engineering Applications of ARTIFICIAL INTELLIGENCE
www.elsevier.com/locate/engappai

Optimizing agent-based meeting scheduling through preference estimation

Andy Chun, Hon Wai*, Rebecca Y.M. Wong

Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong

Received 30 June 2003; received in revised form 16 September 2003; accepted 18 September 2003

Abstract

Meeting scheduling is a routine task that needs to be performed quite regularly and frequently within any organization. Unfortunately, this task can be quite tedious and time-consuming, potentially requiring a several rounds of negotiations among many people on the meeting date, time and place before a meeting can finally be confirmed. The objective of our research is to create an agent-based environment within which meeting scheduling can be performed and optimized. For meeting scheduling, we define optimality as the solution that has the highest average preference level among all the possible choices. Our model tries to mimic real life in that an individual's preferences are not made public. Without complete information, traditional optimal algorithms, such as A* will not work. In this paper, we present a novel "preference estimation" technique that allows us to find optimal solutions to negotiations problems without needing to know the exact preference models of all the meeting participants beforehand. Instead, their preferences are "estimated" and built on the fly based on observations of their responses during negotiation. Another unique contribution is the use of "preference rules" that allow preferences to change dynamical as scheduling decisions are made. This mimics changing preferences as schedule gets filled. This paper uses two negotiation algorithms to compare the effect of "preference estimation"—one that is based on negotiation through *relaxation* and the other that extends this with *preference estimations*. Simulations were then performed to compare these algorithms.

Keywords: Meeting scheduling; Distributed scheduling; Agent-based negotiation; Software agents; User preference modeling

1. Introduction

Meeting scheduling is a common task for organizations of any size. In simple terms, it involves searching for a time and place when and where all the meeting participants are free and available. There may be global (organizational) and local (individual) constraints and preferences on the time and/or location (Yang, 1995). If information was complete, i.e. all the global and local constraints and preferences are known to everyone in the organization, then this can be solved using traditional search algorithms or modeled as a constraint-satisfaction problem (CSP). With complete knowledge, parallel approaches such as artificial neural network (ANN) can also be applied to solving meeting scheduling (Tsuchiya et al., 1996). However, in reality, personal constraints and preferences and even the personal calendar, or part of it, might be hidden from others for privacy. For example, when asked, one might say: "I prefer to have meeting Wednesday or Thursday morning," but is not expected to divulge great details of all the reasons behind this suggestion.

The main focus of our research is in designing an optimizing algorithm that can perform distributed scheduling without complete knowledge of the individual preference models. This is done through a technique we call "preference estimation," which is similar to the concept of "underestimation" in search evaluation functions combined with dynamic programming.

Our agent-based negotiation algorithm works within an environment we call Mobile Agents for Office Automation (MAFOA (Wong et al., 2000)). The scheduling of a person's calendar is performed by negotiations between software agents (Danny and Mitsuru, 1998; Lange and Chang, 1996), called *Secretary Agents* (SA), on behalf of the individuals. Each SA

*Corresponding author. Tel.: +852-278-871-94; fax: +852-278-442-42.
E-mail addresses: andy.chun@cityu.edu.hk (H. Wai), 96476300@alumni.cityu.edu.hk (R.Y.M. Wong).

0952-1976/$ - see front matter © 2003 Published by Elsevier Ltd.
doi:10.1016/j.engappai.2003.09.009

represents one person and has access to that person's calendar, preferences and constraints. This information is hidden from all other agents. By insulating the negotiation process from details of the user model, we enable our negotiation algorithm to work in heterogeneous environments where agents might be built from different agent technologies with different user models and preference strategies. Negotiation in a heterogeneous environment is enabled through a well-defined negotiation protocols and representation standards, such as FIPA's Interaction Protocols (2000) or IETF's iCalendar specification (Dawson and Stenerson, 1998). After a meeting is scheduled, the SA readjusts preferences uses *preference rules*. This simulates real life where preferences might be affected by each decision made.

Our negotiation protocol mimics a relaxation process. The process of proposing and counter proposing a meeting timeslot starts out with everyone putting forward their most preferred timeslot and gradually relaxing their preferences to lesser-preferred slots. A *Meeting Agent* (MA) manages the whole negotiation process. There is one MA per meeting to be scheduled. This mimics real life where there is always a person in charge of scheduling a meeting. The MA is also the agent that performs *preference estimation*. Using the result of *preference estimation*, an MA guides the negotiation to produce an optimal solution. An MA remains active even after the meeting is scheduled to monitor events that might affect the meeting and is deactivated only after the meeting is finally held.

2. State of the art

Our research is related to algorithms for distributed scheduling and agent-based negotiation. Meeting scheduling involves a search through a search space of time intervals constrained by the availability of the participants. In our algorithms, we perform this search as a negotiation process among software agents that act on behalf of the meeting participants. The scheduling in our research is fully automated; scheduling responsibilities are fully delegated to the software agents. Other researchers (Biswas et al., 1992) have investigated distributed meeting scheduling as a computer-assisted process where humans perform negotiation via the computer.

The science and techniques of negotiation have been widely studied in business (Albrecht and Albrecht, 1993) and economics. Research has been performed on different negotiation strategies, communication techniques, models of negotiation, and multiparty negotiations, such as those with mediators or coordinators.

In recent years, automated negotiation has drawn the attention of researchers in computer science and artificial intelligence. This is due partly to commercial interests in using negotiation techniques in the Internet for e-commerce purposes, such as to mimic the role of a salesperson in negotiating a price. Negotiation techniques can also be used for shared resource allocation, data allocation (Schwartz and Kraus, 1997), auctioning, and retail electronic commerce (Guttman and Maes, 1998a). Different negotiation mechanisms (Guttman and Maes, 1998b; Park and Birmingham, 1995) and strategies have been studied.

Defining precisely what negotiation is can be difficult as indicated by Guttman and Maes (1998a). Huhns and Stephens (1999) define agent negotiation as: "Negotiation is a process by which a joint decision is reached by two or more agents, each trying to reach an individual goal or objective. The agents first communicate their positions, which might conflict, and then try to move towards agreement by making concessions or searching for alternatives."

Sen and Durfee (1998, 1996, 1994a) performed a formal study on distributed meeting scheduling (Sen, 1997) and studied the usefulness of different heuristic negotiation strategies to solve distributed meeting scheduling. They have identified different forms of conflicts and commitment strategies that can affect the efficiency of a meeting scheduling system conflict between concurrently active scheduling processes. Sen et al. (1997) presented a user preference mechanism based on a voting scheme. They acknowledge that user preference is important and model preferences as elections between different alternative proposals. Our work is different as we generate alternatives dynamically based on preference estimations that drive the negotiation process to converge towards an optimal result.

Other researchers, such as Sycara et al. also presented a model of distributed negotiation for meeting scheduling. In their approaches, global search is directed by a coordination mechanism, which dynamically passes search control to different agents according to a set of policies. The model gives each one of the participants the flexibility to object if a proposed schedule has low utility for the attendee. User preference is also used in Liu and Sycara (1994) and Garrido and Sycara (1996), each agent specifies a set of meeting preferences for attributes of the meeting, such as date, start-time and length. To maximize individual's meeting preferences, their algorithms schedule the meeting in a calendar interval with attributes closest to its user meeting preferences. In the case of Liu and Sycara (1994) and Garrido and Sycara (1996), it is assumed that the values closest to the preferred one are better. In Liu and Sycara (1994), agents exchange meeting scheduling constraints to create a global model of the problem. In our approach, agents do not have a global model of the problem. They only have knowledge of their own constraints, which seems more realistic. In our system, the meeting coordinator constructs an estimated global

model based responses received from interactions between agents, i.e., *preference estimation*. The actual personal models remain hidden. Although Garrido and Sycara (1996) also keep their preferences private, unlike our approach where there is a centralized meeting coordinator, their meeting scheduling is totally distributed and relies on sharing of partial calendars. Our research tries to mimic real life interactions where there is always one person in charge of scheduling the meeting; otherwise the meeting might not get scheduled at all!

3. Meeting-scheduling problem

The meeting-scheduling problem is a type of negotiation problem. In MAFOA, a negotiation problem is associated with a set of fixed and variable attributes. The initiator of the meeting determines which attributes are fixed and which are variable. Those that are variable will be negotiated. For example, a person calling a meeting might tell his assistant: "I would like to a hold a project meeting sometime next week, preferably next Wed afternoon, with Tom, Nancy,…" In this example, the type of meeting, desired time period when the meeting is to be held, and the attendees are fixed attributes, while the day and time are variable attributes.

The following are examples of meeting attributes:

- *Initiator*: The host or initiator of the meeting. A person might consider a meeting called by his/her immediate supervisor to be more important the others, for example.
- *Rank*: The rank or position of the person calling the meeting. Values can be any rank or position within the organization, such as: CEO, CFO, CTO, VP R&D, VP Sales, etc.
- *Attendees*: The participants or invitees, a list of people that need to attend the meeting. This list may be further classified according to priorities of the attendees, such as those that "must," "should," or "can" attend the meeting. All those that "must attend," must be all available before the meeting can be confirmed, otherwise it will be cancelled. "Should attend" are the normal participants of the meeting. "Can attend" are casual observers; their availability will not affect the meeting schedule.
- *Type*: The type of meeting. For example, values might include: general, departmental, group, strategic, inter-departmental, technical, marketing, sales, project, interview, etc. This value can be used to determine the priority of the meeting during scheduling. Higher-priority meetings might be scheduled earlier or might even take over timeslots from previously scheduled lower priority meetings, i.e., unschedule a meeting, which might get automatically rescheduled by the *Meeting Agent* that is looking after that meeting. Unscheduling is performed using *conflict resolution*.
- *Period*: The time period that the meeting should be held, such as "within the coming 2 weeks," "within this week," "within Friday," etc. The exact date and time is represented by other attributes.
- *Duration*: The length of the meeting. Values may be a number of hours or minutes.
- *Part-of-day*: The part of the day that the meeting will be held. For example, values may be from: breakfast, morning, lunch, afternoon, dinner or evening. This is a coarser grain classification than hours and seems to be more nature in defining time preferences.
- *Day-of-the-week*: Day-of-the-week that the meeting will be held, such as Monday, Tuesday, etc.

3.1. Initiating a meeting

When a person initiates a meeting, he sets up the negotiation problem by deciding which attributes are fixed and which are to be negotiated. For variable attributes, the initiator might decide to further limit the domain of possible values, such as "length" in our example. The initiator constrained the length to be sometime between 2 and 4 h. The exact length of the meeting will still need to be negotiated.

This *negotiation problem* is then handed to that person's SA, who already has the user's preference model. For each new meeting to be scheduled, the SA instantiates a new MA to coordinate the overall negotiation process. The MA also makes negotiation decisions on behalf of the person initiating the meeting.

Negotiation is performed using a negotiation protocol that defines the interactions between agents and also the rules and assumptions of the negotiation process (FIPA Interaction Protocol Library Specification, 2000). Our negotiation protocol is a form of multistage negotiation protocol (Conry et al., 1998) and is a type of contract net protocol (Sen and Durfee, 1996; Smith, 1980). Similar protocols are also used in Sen and Durfee (1998, 1996, 1994a), Sen et al. (1997) and Jennings and Jackson (1995). The content of the negotiation messages can be in a standard knowledge representation, such as KQML (Finin et al., 1997). Fig. 1 outlines a typical sequence of messages involved in MAFOA negotiation:

The following are some of the key messages used in MAFOA negotiation:

- *Announce event*: Details of an event are broadcasted or multicasted asynchronously to all the potential participants of the event. This includes the fixed and variable attributes of the event. Sent along with the announcement is an optional proposal, i.e., set of proposed values for the variable attributes of the event. For example, in meeting scheduling, an event

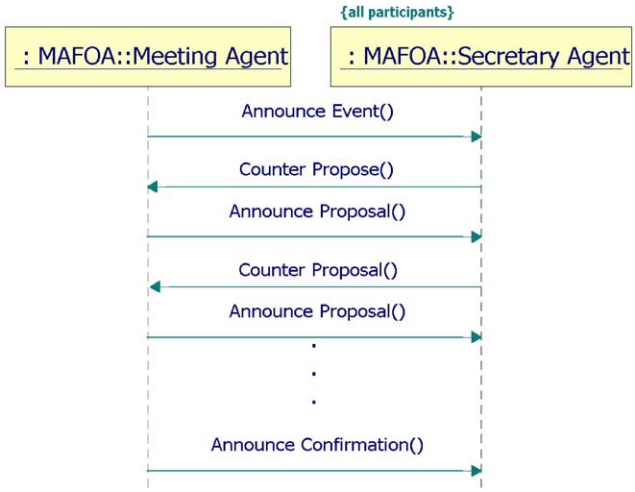

Fig. 1. UML sequence diagram of typical interactions during negotiation.

announcement might be: "Tom would like to hold a departmental meeting next week, preferably next Monday morning."

- *Counter propose*: Once a SA receives a proposal, it will need to give a reply that indicates whether the proposal is acceptable or not and optionally whether it has a counter proposal. For example, a reply might be: "I can make it next Monday morning, but would prefer the afternoon instead." In MAFOA, we assume that all agents are "cooperative" and will reply truthfully on whether a proposal is "acceptable" or not even if the preference level is low. For meeting scheduling, a proposal is "not acceptable" only when a person has a prior committed event that overlaps with the proposed timeslot. We also assume that an agent will always counter propose with its "best" alternative first, which is of course quite logical. Furthermore, we also assume that an SA will not counter propose if the original proposal is already "good enough," i.e., the preference level of the counter proposal must be higher than that of the original proposal. It does not make sense to counter propose with a worse alternative.
- *Announce proposal*: If after processing all the replies and not finding a feasible solution, an MA will generate another proposal that is announced to all the potential participants of the event. For example, "Not everyone can make it next Monday morning, how about Tuesday afternoon instead?" We assume that each new proposal will be slightly worse if not equal in preference level than the previous, from the meeting initiator's point of view.
- *Announce confirmation*: Once a feasible solution has been found, this solution will be announced to all those involved in the event.

4. User preference model

A unique contribution of our research is our formalization of a user preference model that supports optimal negotiation. Studies (Higa et al., 1996) show that fully automated meeting scheduling would only be useful if human factors, such as politics, personal preferences, power structure, etc. are encoded and used by the automated system. Each person has his/her own unique set of personal and business priorities, preferences and constraints. All these come into play during negotiation. In order for us to delegate negotiation to a software agent, it must also understand and be able to make use of the unique preferences of an individual. Our user preference model tries to encapsulate knowledge on different types of personal preferences and how they might change according to changes in the environment. The user preference model is used to influence the behavior of the software agent during negotiation.

In MAFOA, a *negotiation problem* is defined by a finite set of m fixed attributes $f_1, f_2, \ldots, f_m$, whose values will not change during negotiation, and a finite set of n variable attributes $v_1, v_2, \ldots, v_n$, whose values will be negotiated. In addition, each variable attribute v_i is associated with a domain d_i that defines a finite set of possible values $x_1, x_2, \ldots, x_k$, which that variable attribute may be assigned. The value of "time" might be "9 a.m.," "10 a.m.," etc. The user preference model allows users to define priorities on *variable attributes* and preferences on their potential values as well as rules on how these priorities and preferences may change due to changes in the environment or decisions that were made. A negotiation "decision" or solution is defined as a set of assignments of value to variable attributes.

Our user preference model associates *priorities*, *preferences* and *rules* with negotiable variable attributes. Each person may have his/her own set of priorities, preferences and rules, which an agent uses during negotiation to evaluate whether a proposal is acceptable or not and what counter proposals to make. This evaluation results in a *preference level* for each proposal and counter proposal. The following sections provide detail definitions of *priorities*, *preferences*, *rules*, and *preference levels*. This preference model is stored in and accessible only by a person's Secretary Agent.

4.1. Attribute priority

The importance of a negotiable attribute might be different for different people. For example, the "location" of a meeting might be more important than the "time" to a particular person. The *attribute priority* ap_i of a variable attribute v_i defines its degree of importance. It is a number between 0 and $\mathrm{AP}_{\max}$, where $\mathrm{AP}_{\max}$ is a global constant. The importance of a particular variable attribute to a particular person is proportional to the

value of the attribute priority. For example, if the meeting "location" is important to John, then the priority for "location" will be higher than all other attributes of that meeting. Attribute priorities will affect how an agent negotiates and hence influence the outcome of the negotiation process.

To ensure that priorities are used fairly among all the negotiating agents, we normalize the attribute priorities for a given agent such that their total sum must be equal to a fixed global constant AP_{total}.

$$\sum_{x=1}^{n} ap_x = AP_{total}, \tag{1}$$

where n is the total number of variable attributes in the given negotiation problem.

In other words, each agent has the same total number of priority points to use, as it likes, to influence the negotiation, i.e., AP_{total}. This total value will never change and is the same for all the negotiating agents. If a priority is adjusted for an agent, all the priorities of that agent will be affected and normalized back to the same total value. For example, if the user adjusts priorities ap_i to new values ap'_i with a new total value of $AP'_{new} \neq AP_{total}$, then the new normalized priority will be as follows:

$$ap_i = \left(\frac{ap'_i}{AP'_{new}}\right) AP_{total}, \quad \text{where } AP'_{new} = \sum_{x=1}^{n} ap'_x. \tag{2}$$

By default, all variable attributes are considered equally important and hence have the same initial priority value

$$ap_x = AP_{total}/n. \tag{3}$$

4.2. Preference value

A person might consider certain attributes, such as "location," to be more important than others during negotiation. Likewise, he might prefer certain values of these attributes over others, such as preferring location to be in "boardroom" rather than "demo room." We call the amount of preference for a particular value the *preference value*.

For each person, each variable attribute v_j and each potential domain value x_i, there is an associated preference value pv_i. The value of pv_i indicates the preference on having v_j be assigned the value x_i. The degree of "preference" is proportional to the value of pv_i. The value of pv_i may be a number between 0 and PV_{max}, where PV_{max} is a global constant.

For example, the "time" of the meeting may be a variable to be negotiated. A particular user may prefer having the meeting during "lunch." Hence the preference value for "lunch" will be higher than the preference value for any other potential values, such as "morning" or "afternoon."

Furthermore, as a way to ensure that the preferences are not abused or overused by an agent and to give each agent a fair chance in negotiation, we normalize the preference values, such that the preference values $pv_1, pv_2, \ldots, pv_k$, of a variable attribute v_i with domain values $x_1, x_2, \ldots, x_k$ must add up to a fixed global constant PV_{total}.

$$\sum_{x=1}^{k} pv_x = PV_{total}, \tag{4}$$

where k is the total number of domain values for a particular variable attribute.

Initially, by default, there is no special preference on any particular value to be assigned to a variable attribute. Each potential domain value will be treated equally. Hence, all preference values pv_x of that variable attribute v_i will have equal value and be set to

$$pv_x = PV_{total}/k, \tag{5}$$

where k is the total number of values in domain d_i.

4.3. Preference rules

Preference rules is a unique feature of our user preference model that is not found in previous models. In addition to attribute priorities and preference values, each user may also have a finite set of j preference rules $r_1, r_2, \ldots, r_j$, that defines how these priorities and preferences may change as variable attributes are negotiated and assigned values. Each preference rule r_i defines a finite set of conditions that, when all the conditions are satisfied, trigger actions that cause priorities and preferences to be adjusted.

r_i : if $(c_1, c_2, \ldots, c_j)$ then $(a_1, a_2, \ldots, a_k)$.

In MAFOA, rule conditions $c_1, c_2, \ldots, c_j$ are defined as a pattern of values on fixed attributes, variable attributes or global attributes (attributes that are shared by all agents). Each c_i defines a pattern that will be matched against scheduled events. For example, a rule might be "If there is already a meeting on Monday morning, then I don't want any other meetings in the afternoon." The condition of "meeting on Monday morning" defines a pattern that will be matched against all scheduled events.

The rule consequence consists of a finite set of actions $a_1, a_2, \ldots, a_k$ to be performed on priorities and preferences once the rule conditions are satisfied. In MAFOA, we have defined four possible actions on attribute priorities:

- incAP(): To increment the attribute priority ap_i of a given variable attribute v_j by a given amount *delta*. All the attribute priorities of the given agent will then be normalized back to the constant sum of AP_{total} using Eq. (2).

- decAP(): To decrement the attribute priority ap_i of a given variable attribute v_j by a given amount *delta* and then perform normalization.
- max AP(): To assign the maximum possible attribute priority $ap_i = AP_{max}$ to the given variable attribute v_j; all other priorities of this agent will be set to zero for normalization.
- min AP(): To assign an attribute priority $ap_i = 0$ to the given variable attribute v_j and then perform normalization.

There are another four related actions that can be performed on preference values:

- incPV(): To increment the preference value pv_i of a given value x_i of a particular variable attribute v_j by a given amount *delta*. All the preference values associated with v_j will then be normalized back to the constant sum of PV_{total}.
- decPV(): To decrement the preference value pv_i of a given value x_i of a particular variable attribute v_j by a given amount *delta* and then perform normalization.
- max PV(): To assign the maximum possible preference value $pv_i = PV_{max}$ to the given value x_i of a particular variable attribute v_j; all other preference values of this variable attribute will be set to zero for normalization.
- min PV(): To assign a preference value $pv_i = 0$ to the given value x_i of a particular variable attribute v_j and then perform normalization.

Our *preference rule* technique makes our user preference model more realistic when compared with previous models, which are all static. Our model is dynamic and automatically changes and adopts itself with the current environment. Although the model is dynamic, preference rules are checked and fired only after each scheduling decision. They are not fired during the dynamic negotiation process—as nothing in the schedule has changed to cause their preferences to change. For example, if a meeting has just been scheduled for Friday afternoon, this will impact the preference for scheduling another meeting in the same morning. Therefore, the preference rules will not impact the optimality of the solution, as the preference values will remain constant until a decision point is reached.

4.4. Preference levels

In MAFOA, a potential solution to a negotiation problem is defined as a tuple from $d_1xd_2x\ldots xd_n$ such that the n assignments of values to the problem's variable attributes is to the "satisfaction" of all the negotiating agents, i.e., a compromise is found. During the negotiation process, each negotiating agent will need to evaluate how "satisfied" it might or might not be with the proposed compromise/solution. In MAFOA, each proposed solution is called a *proposal* when offered by the *initiating* agent, i.e., the agent that initiated the problem to be negotiated, and a *counter proposal* when offered by any other agent involved in the negotiation.

For example, a proposal/counter proposal P_x might be the tuple $(V_1, V_2, \ldots V_n)$ where each V_j is a constant value from the domain d_j of variable attribute v_j. If we need to negotiate the "day" and "time" of a meeting, a potential proposal might be the tuple ("Tue", "9 a.m.").

In MAFOA, the result of evaluating how satisfied an agent is with a proposal or counter proposal is called the *preference level* of that proposal. Different agents might of course potentially have a different preference level for the same proposal. For agent i, for a negotiation problem with n variable attributes, the preference level pl_i for a particular proposal/counter proposal P_x is defined as

$$pl_i(P_x) = \sum_{j=1}^{n} ap_j\, pv_{jk}, \tag{5}$$

where pv_{jk} is the preference value for the assignment of value V_k to the variable attribute v_j and ap_j is the attribute priority.

A proposal/counter proposal with a higher preference level means it is more preferred.

4.5. Defining priorities and preferences

In MAFOA, a person defines and modifies his/her user preference model via a simple interface, such as that shown in Fig. 2. Pino and Mora (1997) and Pino et al. (1998) offers a different form of graphic user interfaces based on rules and timeslots to define meeting preferences, which they call lateral model. In their case, a human meeting coordinator uses the preference information. In MAFOA, only an individual's Secretary Agent uses the user preference model.

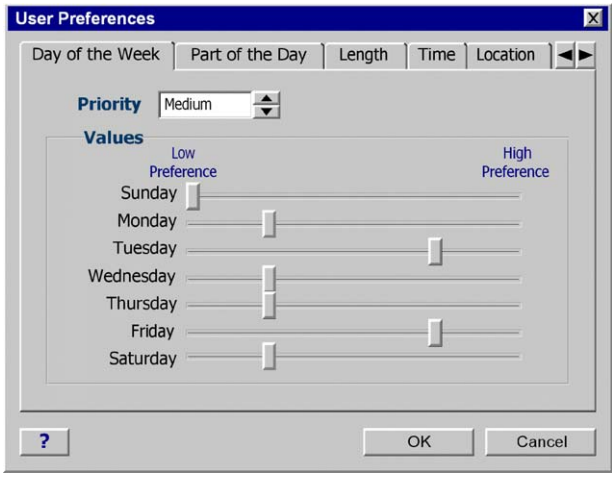

Fig. 2. A menu to adjust user preferences for "day of the week" attribute.

In MAFOA, for each attribute, the user can assign a priority. For each potential attribute value, the user can assign a preference. The details of the mathematics involved are hidden from the user. Once he confirms his preferences, the system automatically normalizes all the values according to normalization rules defined above. Luo et al. (2000) takes an alternative approach and use fuzzy constraints to model preferences. However, fuzzy preferences must be defined for individual timeslots. In our model, preferences can be define much more easily and is adequate for the purpose of determining a *preference level* for a proposal.

5. Agent structure

For meeting scheduling, MAFOA uses the services of two types of software agents—MA and SA. There is one MA per meeting and there is one SA per user.

5.1. Secretary agent

An SA, like its human counterpart, knows a person's individual calendar or schedule, possibly both business and personal, as well as that person's preferences. This schedule is stored in a "Schedule" data structure that is basically a timetable of events that a personal is committed to participating and events that a person is interested in and might consider participating if time permits, and general broadcasted events. A person's preferences are stored using the "User Preference Model" we have defined earlier that consists of priorities, preferences and rules.

Given a person's schedule and preferences, we will then be able to determine when a person is free and what his preferences are in using those free timeslots for a particular event, such as a meeting. For each *announced* event that the user is invited to participate, the SA will create a sorted "preference vector" (PV) (see Fig. 3) to store pre-computed preferences to be used during negotiation in evaluating proposals and in generating counter proposals.

5.2. Preference vector

The *preference vector* is a sorted vector of a person's preferences on different alternative proposals for an announced event, such as a meeting. The vector is sorted according to the value of the *preference level* of a proposed alternative. The preference level is computed based on Eq. (5), which was explained earlier in the paper. For example, if we have a simplified meeting event with only two attributes—day-of-the-week and part-of-the-day, then "Mon at noon," "Tue in the morning," "Fri afternoon," etc. would be valid alternatives or proposals. The preference level for these proposals will be different for different people and will depend on parameters stored in the user preference model. For meeting scheduling, only proposals that the user is willing to commit to are included, i.e., only proposals for time periods when the user is free will be included.

The PV contains a "current" pointer (Fig. 4) that points to the current counter proposal that has been offered by this SA to a MA. An SA will start off by offering what it considers to be the "best" counter proposal for its user, i.e., the counter proposal with the highest preference level, and gradually move down the vector. The difference between the final committed proposal and the best represents the "compromise" that this SA has made.

5.3. Meeting agent

The role of an MA is to coordinate the scheduling of an individual meeting event, to track global events that might affect that meeting schedule and decommissioning itself when the meeting is finally held. An MA also represents the interest of the user that initiated the meeting when negotiation with other agents. An SA creates an MA when the user initiates a meeting. The SA also passes along the PV to the MA. Only an SA can generate a PV, since the user preference model is only accessible by the SA.

The MA uses the PV to generate proposals that is offered to each SA involved in the meeting. In return, each SA can firstly accept or refuse the proposal, and

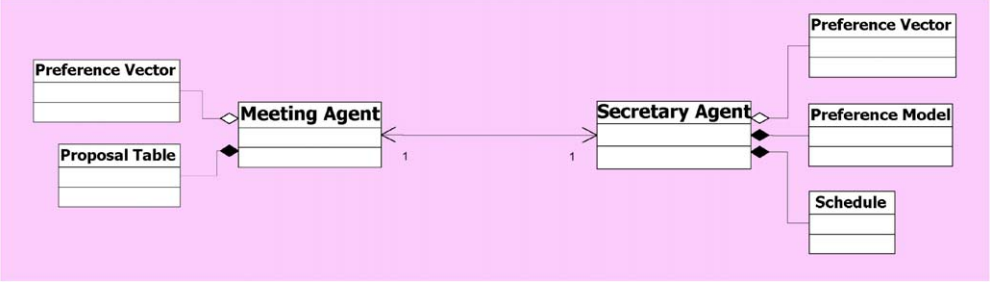

Fig. 3. UML class diagram of the MA and SA.

secondly offer a counter proposal from its our PV. These proposals and counter proposals are all stored and tallied in a "Proposal Table" data structure (Fig. 5) that allows a MA to make negotiation decisions.

A "timeslot" in the proposal table is a combination of "start time" and "duration," as there may be proposals with different durations as well as start times. There is a marker or token (see Fig. 5) when a timeslot has been proposed. If a participating SA accepts this timeslot, another token will be added. For each set of proposals announced by the MA, the SA can optionally offer a set of counter proposals. These counter proposals are also represented by tokens and stored in their respective timeslots. If any SA rejects a proposal, that timeslot will be marked as *infeasible* and will not be considered anymore, for example timeslot T4 in Fig. 5. A solution is found when there is a timeslot where the total number of tokens equals the number of participants.

The purpose of the proposal table is to provide additional information to an MA to help it make decisions on negotiation strategies. For example, it may decide to go for a quick solution by offering proposals with high "mass appeal," i.e., those timeslots with high number of tokens, such as timeslot T2 in Fig. 6. Or, it may decide to go for a high quality solution (from the meeting initiator's point of view) by continuing to propose from its PV.

6. The negotiation algorithms

A unique contribution of our research is the use of *preference estimations* that allows optimal meeting schedules to be found—schedules with highest average preference levels. To illustrate how *preference estimation* works, this paper compares two negotiation algorithms: Algorithm 1 (NWOPI), which is based on *relaxation* only, and Algorithm 2 (NWPI), which is based on *relaxation* plus *preference estimations*.

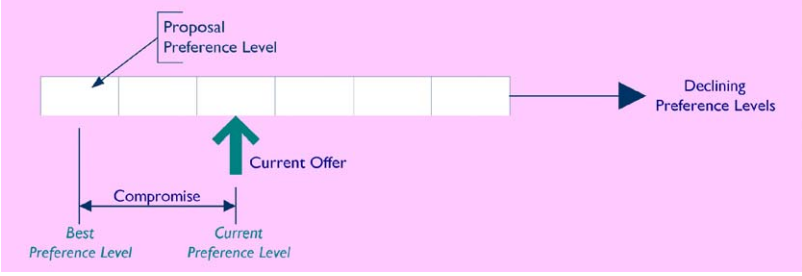

Fig. 4. The structure of the PV.

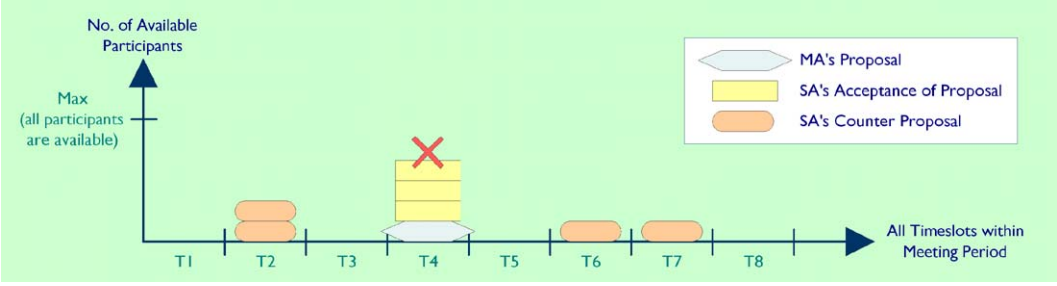

Fig. 5. The structure of the *proposal table*.

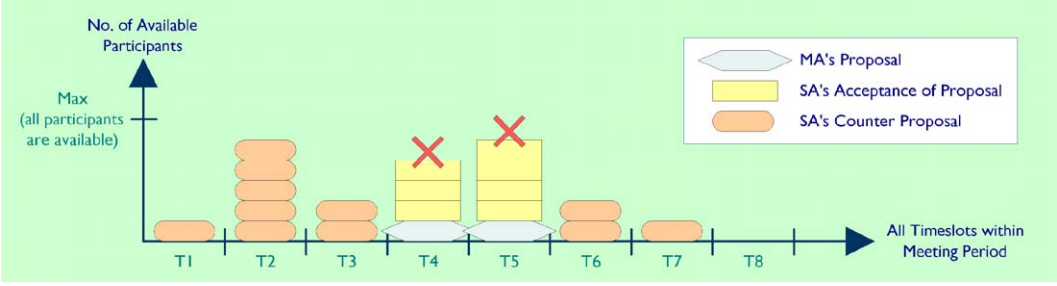

Fig. 6. Example of the proposal table after some more negotiation.

6.1. Relaxation

Our relaxation-based negotiation is in some ways similar to performing heuristic-guided search. The heuristics determines which potential solutions are more preferred over others within an upper and lower bound in terms of *preference levels*. The heuristics or preferences guide the search in locating higher quality solutions. However, preferences are unique to each individual. Therefore, each agent's heuristics might contradict and compete with each other. If everyone holds on to their most preferred proposal, the problem will be over constrained and there will be no solution. The relaxation process gradually lowers the lower bounds on the preference level and expands the search tree until a mutually beneficial solution is found.

Initially, differences between agents may be great. A proposal that is good for one agent might be really bad for another. The relaxation process gradually reduces this difference until a common ground or compromise is found. The MA, responsible for coordinating the negotiation process, will use proposals stored in its PV to lead the relaxation process.

Bui et al. (1995) offers a different approach to negotiation called *incremental negotiation* where negotiation are performed on meeting attributes in a hierarchical structure. For example, the part of the week (early or late) might first be negotiated, then the day of the week, and then the hour. In other words, individual variable attributes are negotiated one after another in hierarchical fashion. In MAFAO, a proposal contains a complete set of proposed variable attribute values and the whole set is negotiated.

6.2. Preference estimation

Since the negotiation or distributed search process is guided by MA's proposals, the speed and accuracy with which a solution is found depends on whether the MA can propose solutions that everyone might be "happy" with and early on in the search. This ability will reduce the number of negotiation cycles that the MA must go through. However, since an individual's user preference model is hidden from others, *preference estimation* is a technique to dynamically "guess" global preferences in generating proposals without needing details of each of the participant's preference model. Details of preference estimation will be described later in Algorithm 2 (NWPI). This is similar to how a human negotiator would try to guess what the other values as more important to come up with a compromise solution.

6.3. Algorithm 1 (NWOPI)—relaxation

Algorithm 1 (NWOPI) is to highlight how the relaxation technique is used to guide a negotiation. Algorithm 1 is similar to the negotiation process defined in Sen and Durfee (1998). In Sen and Durfee (1998), the negotiation process uses the "earliest" timeslot to guide the negotiation. This means timeslots will be proposed using an "earliest available timeslot first" heuristic. In our approach, the user preference model guides the search—the timeslot with the "best" preference estimations first. We believe this is a better approach, as the "earliest" timeslot might not necessarily always be the most preferred one. In Algorithm 1, individual SA's preferences are not passed back to the MA. This is similar to the *private preference* model of Garrido and Sycara (1996).

Fig. 7 highlights the main flow of the MAFOA negotiation algorithms. Both Algorithms 1 and 2 follow basically the same flow—negotiation consists of iterations of proposing and counter proposing. The main difference is in the type of information stored in the counter proposal as well as how new proposals are generated. For Algorithm 1, individual's preferences are hidden from the MA. The MA relies mainly on the preferences of the initiating agent plus the counter proposals to guide the search. For Algorithm 2, preference levels are fed back to the MA, which allows it to form preference estimations and guide the negotiation towards an optimal solution.

The key action states in our algorithms are:

Step 1: Define meeting—the user initiating the meeting provides details of the meeting to be scheduled and then submits them to his SA. For each meeting to be scheduled, the SA creates a MA to manage the negotiation process.

Step 2: Generate preferences—the SA, based on the person's calendar and user preference model, produces a list of possible proposals, which are then sorted according to their preference levels and stored in a PV. This PV will then be passed to the newly created MA.

Step 3: Initialization—the MA initializes itself with the SA's PV. It also starts the actions 4.1x and 4.2, which are performed in parallel.

Steps 4.1x:

(a) *Select proposal*—For Algorithm 1, we use a "best first" strategy to select the "best" proposals, i.e., the proposal with the highest preference level, from the MA's PV.
(b) *Announce meeting*—Details of the meeting plus n selected proposals will be announced to the SA of each meeting participant.
(c) *Generate preferences*—This is basically the same as Step 2 but for the participants. This PV generated will not be passed to the MA. It will only be stored within and known to the individual SA.
(d) *Counter propose*—The participant SA first determines whether any of the n proposals are feasible or not. It then tries to select m counter proposals that

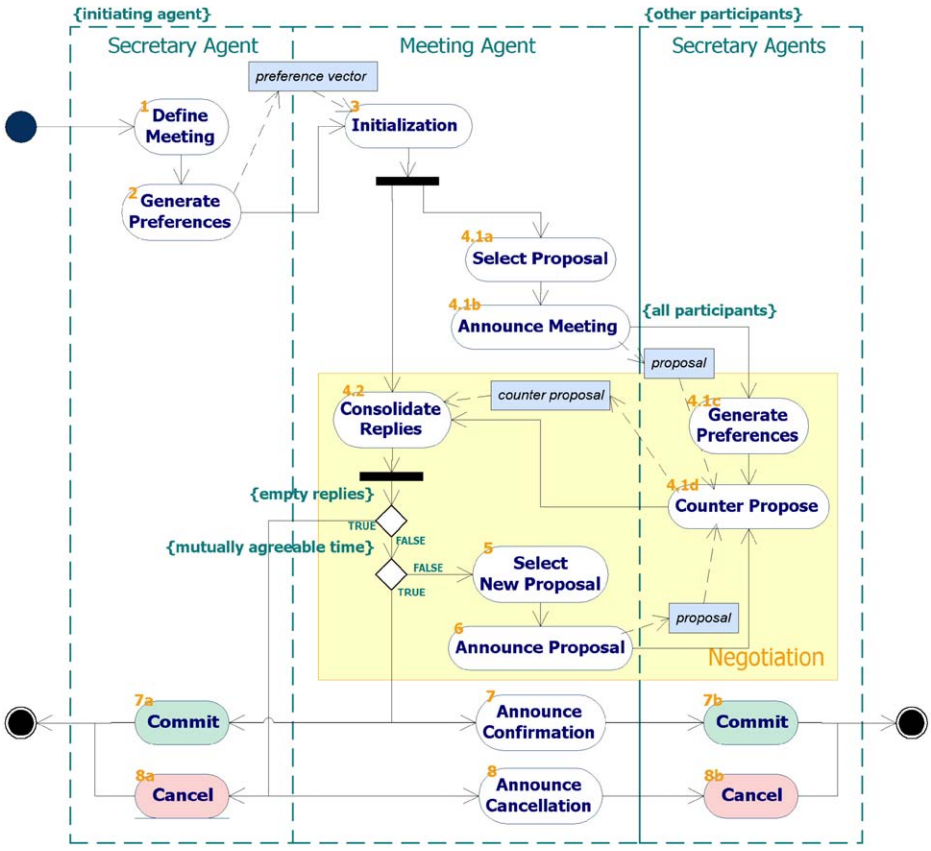

Fig. 7. UML activity diagram of the negotiation algorithms.

have higher preference levels than those proposed by the MA. It passes both sets of information as a "counter proposal" to the MA.

Step 4.2: Consolidate replies—the MA collects these counter proposals from all the participating SAs and consolidates them to determine which of the following three cases to follow:

Case 1: No solution—if any one SA returns an "empty" counter proposal—none of the n proposals are acceptable and no counter proposal—then there is no solution and the meeting is cancelled, i.e., go to Step 8.

Case 2: Proposal accepted!—otherwise, it tries to determine whether a consensus has been made, i.e., if all the participants of the meeting accept at least one of the n proposals. If so, it goes to Step 7 to announce that the meeting is confirmed.

Case 3: Proposal not accepted—If none of the n proposals are accepted, the MA must make select and announce another set of n proposals, i.e., go to Step 5.

Step 5: Select new proposal—this step uses the same strategy as Step 4.1a in selecting the next set of n proposals to announce.

Step 6: Announce proposal—this is similar to Step 4.1b except that only the proposals are announced; without the details of the meeting, which are already stored locally in each SA.

Step 7: Announce confirmation—an announcement is made to all the meeting participants that a proposal has been mutually accepted. The SA shall commit this in its calendar of events.

Step 8: Announce cancellation—an announcement is made to all the meeting participants that the meeting cannot be scheduled and is cancelled.

6.4. Algorithm 2 (NWPI)—relaxation+preference estimation

Algorithm 2 (NWPI) extends Algorithm 1 with what we call *preference estimation* that allows an MA to "estimate" the global preference level of a proposal and then select those proposals that are more likely to get accepted, hence generating higher quality solutions. In order to do so, individual SA's preferences for each proposal/counter proposal are also sent to the MA with their replies. This is similar to the *public preference* model of Garrido and Sycara (1996). In MAFOA, we make use of a *preference estimation vector* (PEV) to perform preference estimation.

The PEV is similar to the PV (see Fig. 4) in that it is a sorted vector of proposals and preference levels. However, unlike the PV where the preference level is based purely on the user preference model of a single user, the PEV consolidates preference levels from all the

participating agents and is dynamically sorted according to the "estimated" global preference level at that time. It is only "estimated" since the MA does not have access to the actual user preference models of the individual meeting participants.

Preference estimation in Algorithm 2 is done by extending three of the steps in Algorithm 1—Steps 4.1d, 4.2 and 5.

Step 1: Define meeting—initialize negotiation problem.

Step 2: Generate preferences—initiator's SA generates preferences for newly created MA.

Step 3: Initialization—the MA initializes itself and performs 4.1x and 4.2 in parallel.

Steps 4.1x:

(a) Select proposal—select next n "best" proposals.
(b) Announce meeting—announce meeting and n selected proposals.
(c) Generate preferences—meeting participants generate preferences.
(d) Counter propose—this is similar to Algorithm 1, except that proposal confirmation and counter proposals are all sent back together with the actual *preference level*; the user preference model is never passed to anyone. This is similar to Jennings and Jackson (1995) where an average preference level is computed from the individual preference levels that are passed back. However, we believe our approach is better than that in Jennings and Jackson (1995), since we use preference estimation to reduce the number of interactions needed to find an optimal solution. In Jennings and Jackson (1995), an average preference level is computed from the individual preference levels that are passed back. However, Jennings and Jackson (1995) does not use preference estimation to rank the remaining proposals. Algorithm 1 simulates the approach used in Jennings and Jackson (1995), while Algorithm 2 shows that with the addition of *preference estimation* optimal schedules can be produced. Since counter proposals' preference levels are always decreasing, i.e., always propose more preferred alternatives first; we can use these values as an upper bound on any future preference levels for this SA. Fig. 8 shows the proposal table that consolidates counter proposals, now with preference levels.

Algorithm 2 also contains a "latest preference list" (LPL) that stores the "most recent" preference level received from each of the meeting participants. These values become the upper bounds on preference levels for any future counter proposals from the participants.

Step 4.2: Consolidate replies—the end conditions for Algorithm 2 are slightly different from that of Algorithm 1. Algorithm 1 stops as soon as a "common interval" or consensus has been found. Algorithm 2, however, continues the search until the "common interval" found is the one with the highest average preference level, i.e., the first element in the PEV. Similar to branch-and-bound (Lawler and Wood, 1966) or A* search (Shapiro, 1992), the search does not stop, even after a solution has been found, until all other alternatives have poorer evaluation scores. In our case, the average preference level of all other alternatives must be lower than that of the newly found "common interval." After receiving all the counter proposals, the PEV is updated with more accurate preference levels and resorted. Based on the new PEV, Algorithm 2's end conditions are:

Case 1: No solution—if any one SA returns an "empty" counter proposal—none of the n proposals are acceptable and no counter proposal—then there is no solution and the meeting is cancelled, i.e., go to Step 8.

Case 2: Proposal accepted—otherwise, it tries to determine whether a consensus has been made, i.e., if all the participants of the meeting accept at least one of the n proposals. If so and if this consensus or "common interval" has the best average preference level, it goes to Step 7 to announce that the meeting is confirmed.

Case 3: Proposal not accepted—otherwise, the MA must make select and announce another set of n proposals, i.e., go to Step 5.

Step 5: Select new proposal—instead of selecting the "best" n proposals from the PV, Algorithm 2 selects the n best proposals from the dynamically sorted PEV, which consolidates preference levels received from Step 4.1d. Since the negotiation process will be directly driven from the PEV, this is a form of dynamic programming

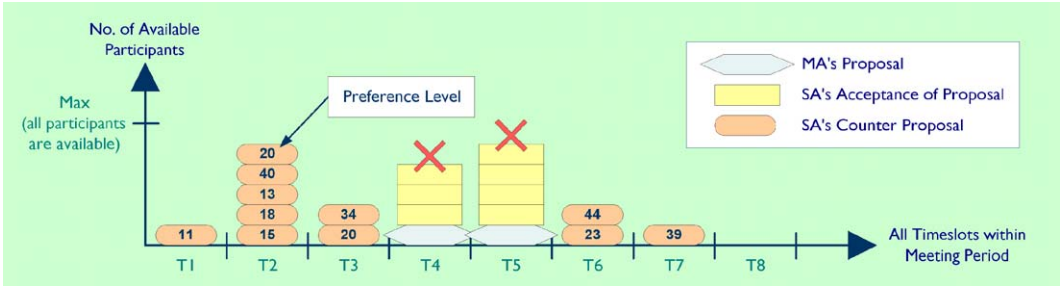

Fig. 8. The proposal table for Algorithm 2 with additional preference levels.

that allows the MA to dynamically change negotiation direction as more information is received through counter proposals.

Step 6: Announce proposal—announce n proposals to participants.

Step 7: Announce confirmation—confirm meeting.

Step 8: Announce cancellation—cancel meeting.

The PEV is dynamically sorted using the average "estimated" *preference level* of each proposal/counter proposal. Only the MA has the PEV, as the MA is responsible for driving the negotiation process through its proposal selection from the PEV. For agent i, the estimated preference level $\mathrm{pl}'_i(P_x)$ of any proposal/counter proposal P_x is defined as

$$\mathrm{pl}'_i(P_x) = \begin{cases} \mathrm{pl}_i(P_x) \text{ if } P_x \text{ has already been proposed/} \\ \quad \text{counter proposed,} \\ \min(\mathrm{pl}_i(P_x)|_x) \text{ for all } P_x \text{ proposed/} \\ \quad \text{counter proposed.} \end{cases} \tag{6}$$

The *estimated preference level* $\mathrm{pl}'_i(P_x)$ is equal to the actual $\mathrm{pl}_i(P_x)$ if P_x has already been proposed or counter proposed before. As Step 4.1d requires the actual preference level to be passed back to the MA during the counter propose step. Otherwise, we will use the minimum preference level received so far from the SA. This is equivalent to the "most recent" preference level, as the value will be decreasing. SA will always propose the "best" or "most preferred" alternatives first. This usage of the minimum value is similar to "underestimation" in A* search algorithms (Shapiro, 1992). The evaluation function $f'(n)$ in A* is equal to $g(n) + h'(n)$, where $g(n)$ is the actual cost from start to node n and $h'(n)$ is the estimated minimal cost path from n to goal, i.e., the underestimated remaining cost. For our meeting-scheduling algorithm, n is analogous to a particular cycle in the negotiation process. The actual costs are the preference levels of the counter proposals received and the estimated costs are the estimated preference levels of new proposals. By using the "current" or "minimum" preference level as the estimated preference level, we guarantee that good solutions will not be "overlooked;" since the actual preference level will in fact be lower.

Instead of adding the costs, our evaluation function computes the *average preference level*. With the $\mathrm{pl}'_i(P_x)$ value for every participant and every proposal, the MA can then compute the global *average preference level* $\mathrm{apl}(P_x)$ for each proposal P_x:

$$\mathrm{apl}(P_x) = \frac{\sum_{i=1}^{n} \mathrm{pl}'_i(P_x)}{n}, \tag{7}$$

where n is the total number of participants.

This *average preference level* provides a more "global" indication of how preferred one proposal might be compared with another and hence the chances of it getting accepted. The average preference level $\mathrm{apl}(P_x)$ for each proposal is updated once per negotiation cycle to reflect additional preference level information received during the negotiation, i.e., the $\mathrm{pl}'_i(P_x)$ value will get more and more precise with each negotiation cycle. The PEV is dynamically sorted in Step 4.2 prior to each proposal selection step (Step 5). In Algorithm 2, the MA will propose the n proposals with the highest $\mathrm{apl}(P)$ value during each cycle.

For some negotiation problems, different levels of collaboration may be required or expected from different participating agents. For example, there may be a group of agents that "must" participate in the event, a group that "should" participate, and a group that "can" participate if they so wish. For these cases, a weighted evaluation function is used to give more consideration to preferences from the "must" and "should" group:

$$f(P_x) = \frac{\sum_i^{\text{must}} w_m \mathrm{pl}'_i(P_x) + \sum_j^{\text{should}} w_s \mathrm{pl}'_j(P_x) + \sum_k^{\text{can}} w_c \mathrm{pl}'_k(P_x)}{n}, \tag{8}$$

where n is the total number of participants in the event; i is an agent from the "must" group, j is an agent from the "should" group, k is an agent from the "can" group, w_m is a weight factor for the "must" group and is usually greater than 1.0, w_s is a weight factor for the "should" group and is usually greater than or equal to 1.0 and w_c is a weight factor for the "can" group and is usually less than or equal to 1.0.

7. Computer simulations

Computer simulation was performed for the following objectives: to compare the performances of Algorithms 1 and 2, verify that Algorithm 2 produces optimal solutions, and to determine the tradeoffs in using Algorithm 1 versus 2. In our simulation program, we are able to specific a set of simulation parameters:

Number of simulation cycles—total number of times the whole simulation is to be executed.

Hours of meetings (H)—total number of hours of meetings to be scheduled/simulation cycle.

Max number of participants—maximum number of participants in any one meeting.

Calendar size—the total number of days of meeting scheduling to be simulated.

Length of day—the number of hours in a day to be scheduled. For example, we might just want to simulate the scheduling of meetings within office hours.

Calendar density (CD)—the number of hours that is already preoccupied by prior engagements. This is to simulate the effect of having some timeslots

pre-allocated from previous meeting scheduling exercises or for annual leave, training, holidays, etc.

Number of proposals (n)—number of proposals sent from MA to SA during each cycle.

Number of counter proposals (m)—number of counter proposals to be returned from SA to MA during each negotiation cycle.

The initial state of each attendee's calendar is determined by the parameter "calendar density" (CD) that indicates the number of occupied slot in the attendee's calendar. The exact occupied slots are also randomly generated, i.e. if CD = 5, the program will randomly generate 5 busy slots in each person's calendar using a normal distribution. Each randomly generated meeting event is defined by a set of attributes—the start-time, end-time, length, participants, host, etc. For simulation purposes, meeting lengths are randomly generated. The "attribute priority" and "preference value" are also randomly generated for each participant for each simulation run. Since this paper does not cover commitment strategies, our computer simulation schedules the randomly generated meetings one at a time. Furthermore, we assume that the meetings are independent events, i.e., no sequential constraints are imposed. Detailed analysis on the effects of different commitment strategies for concurrent scheduling of meetings were carried out in Sen and Durfee (1994b). In our simulations, since there is no concurrency, we can use either the *committed strategy* or the *non-committed strategy* as defined in Sen and Durfee (1994b). The "preference rules" are not involved in the simulation as they are only used after a meeting has been scheduled and do not affect the outcome of the simulations.

For the purpose of comparison, we defined the following measurements to be made on the simulation results. The basic measurements are:

H number of hours to be scheduled;

CD number of occupied hours in the calendar per agent;

S_{HC} set of randomly generated meeting based on parameters H and CD;

$|S_{\mathrm{HC}}|$ number of meetings in S_{HC};

$|S_{i\mathrm{HC}}|$ number of meeting in S_{HC} that involves agent i

$M_{j\mathrm{HC}}$ jth meeting in S_{HC} to be scheduled (contains many alternative proposals);

$PPt_{j\mathrm{HC}}$ number of participant in $M_{j\mathrm{HC}}$;

$R_{j\mathrm{HC}}$ total number of requests required to schedule meeting $M_{j\mathrm{HC}}$;

R_{HC} average number of requests to schedule meetings in the meeting set S_{HC}

$$R_{\mathrm{HC}} = \left(\sum_{1}^{j} R_{j\mathrm{HC}}\right)/|S_{\mathrm{HC}}|.$$

Measurements related to committed meetings:

$P_{ij\mathrm{HC}}$ agent i's preference level for the final committed proposal for $M_{j\mathrm{HC}}$.;

$\mathrm{AP}_{j\mathrm{HC}}$ average preference level (AP) for committed meeting $M_{j\mathrm{HC}}$

$$\mathrm{AP}_{j\mathrm{HC}} = \left(\sum_{1}^{i} P_{ij\mathrm{HC}}\right)/PPt_{j\mathrm{HC}}.$$

$\mathrm{AP}_{\mathrm{HC}}$ average of the AP's for all the committed meetings in set S_{HC}

$$\mathrm{AP}_{\mathrm{HC}} = \left(\sum_{1}^{i} \mathrm{AP}_{j\mathrm{HC}}\right)/|\mathrm{S}_{\mathrm{HC}}|.$$

Measurements related to overall optimal solutions:

$O_{j\mathrm{HC}}$ optimal or best AP for all proposals in meeting $M_{j\mathrm{HC}}$;

$\mathrm{AO}_{\mathrm{HC}}$ average of all the optimal AP's for the whole meeting set S_{HC};

$$\mathrm{AO}_{\mathrm{HC}} = \left(\sum^{j} O_{j\mathrm{HC}}\right)/|S_{\mathrm{HC}}|.$$

$\mathrm{DO}_{j\mathrm{HC}}$ difference between optimal AP and AP of committed proposal for $M_{j\mathrm{HC}}$

$$\mathrm{ADO}_{j\mathrm{HC}} = O_{j\mathrm{HC}} - \mathrm{AP}_{j\mathrm{HC}}.$$

$\mathrm{ADO}_{\mathrm{HC}}$ average difference between optimal AP and committed AP for S_{HC}

$$\mathrm{ADO}_{\mathrm{HC}} = \left(\sum^{j} \mathrm{ADO}_{j\mathrm{HC}}\right)/|S_{\mathrm{HC}}|.$$

Measurements related to individual's "expected" solutions:

$\mathrm{EP}_{ij\mathrm{HC}}$ agent i's "expected" preference for meeting $M_{j\mathrm{HC}}$—the highest preference level out of all the possible proposals for $M_{j\mathrm{HC}}$;

$\mathrm{DE}_{ij\mathrm{HC}}$ agent i's difference between expected and final preference level for $M_{j\mathrm{HC}}$.

$$\mathrm{DE}_{ij\mathrm{HC}} = \mathrm{EP}_{ij\mathrm{HC}} - P_{ij\mathrm{HC}}.$$

$\mathrm{ADE}_{j\mathrm{HC}}$ average difference from expectation on meeting $M_{j\mathrm{HC}}$

$$\mathrm{ADE}_{j\mathrm{HC}} = \left(\sum^{i} \mathrm{DE}_{ij\mathrm{HC}}\right)/PPt_{j\mathrm{HC}}.$$

$\mathrm{ADE}_{\mathrm{HC}}$ average difference from expectation on whole meeting set S_{HC}

$$\mathrm{ADE}_{\mathrm{HC}} = \left(\sum^{j} \mathrm{ADE}_{ij\mathrm{HC}}\right)/|S_{\mathrm{HC}}|.$$

$\mathrm{PD}_{ij\mathrm{HC}}$ percentage deviation of actual vs. expected preference level for agent i on meeting $M_{j\mathrm{HC}}$

$$\mathrm{PD}_{ij\mathrm{HC}} = \mathrm{DE}_{ij\mathrm{HC}}/\mathrm{EP}_{ij\mathrm{HC}} \times 100\%.$$

$\mathrm{APD}_{i\mathrm{HC}}$ average percentage deviation of agent i on meeting set S_{HC}

$$\mathrm{APD}_{i\mathrm{HC}} = \left(\sum^{j} \mathrm{PD}_{ij\mathrm{HC}}\right)/|S_{i\mathrm{HC}}|.$$

$\mathrm{PD}_{j\mathrm{HC}}$ average percentage deviation for meeting $M_{j\mathrm{HC}}$ for all participants

$$\mathrm{PD}_{j\mathrm{HC}} = \left(\sum^{j} \mathrm{PD}_{ij\mathrm{HC}}\right)/\mathrm{PPt}_{j\mathrm{HC}}.$$

$\mathrm{APD}_{\mathrm{HC}}$ average percentage deviation on the meeting set S_{HC}

$$\mathrm{APD}_{\mathrm{HC}} = \left(\sum^{j} \mathrm{PD}_{j\mathrm{HC}}\right)/|S_{\mathrm{HC}}|.$$

7.1. Simulation results

This section compares the simulation results of Algorithm 1 with Algorithm 2 from executing our simulation program with the following simulation parameters. (The simulation setting used by our experiments was selected to simulate those used in Sen and Durfee, 1998).

Number of simulation cycles = 100;
Hours of meetings = 35;
Max number of participants = 6;
Calendar size = 6;
Length of day = 8;
Calendar density (CD) = from 0 to 13;
Number of proposals = 1;
Number of counter proposals = 1.

Table 1 summarizes a few key measurements for Algorithm 1 (NWOPI) and Algorithm 2 (NWPI) with average results obtained over 100 simulation runs with input parameter "hours of meetings" = 35 and "calendar density" ranging from 0 to 13. The average optimal preference level, $\mathrm{AO}_{\mathrm{HC}}$, and the average difference from optimal, $\mathrm{ADO}_{\mathrm{HC}}$, of the two algorithms in different calendar densities are shown in Table 1. The $\mathrm{AO}_{\mathrm{HC}}$ is the average of the highest possible preference level that can be found for each meeting to be scheduled. This value is slightly lower for Algorithm 2, as it always finds an optimal solution, hence the optimal preference value for each next meeting to be scheduled will be lower. For Algorithm 1, the objective is to find a quick solution without optimality. The possible optimal value will always be higher as the preferred time slots might not have been taken. As the table shows, Algorithm 2 (NWPI) always finds an optimal solution. This is an interesting result, as past research in negotiated meeting scheduling only focused on modeling or algorithm efficiency, but not optimality. This result shows that optimality can be obtained by the simple addition of *preference estimation*.

Chart 1 shows that the average preference levels of the committed meeting schedules obtained by Algorithms 1 and 2. Obviously, the overall preference level will

Table 1
Optimal preference level and average differences from optimal

Calendar density	$\mathrm{AO}_{\mathrm{HC}}$		$\mathrm{ADO}_{\mathrm{HC}}$	
	Alg 1 NWOPI	Alg 2 NWPI	Alg 1 NWOPI	Alg 2 NWPI
0	28.82753	28.144682	2.173903	0
1	28.13506	27.522175	2.075616	0
2	27.37358	26.7121	1.986882	0
3	26.40423	25.933998	1.892491	0
4	25.7022	24.976007	1.85187	0
5	24.72347	24.100136	1.677021	0
6	23.87449	23.418562	1.571699	0
7	22.87245	22.53191	1.480634	0
8	22.04794	21.933178	1.385773	0
9	21.2655	21.050583	1.22031	0
10	20.61108	20.303194	1.239456	0
11	19.95342	19.65437	1.199134	0
12	18.98034	18.683361	1.053649	0
13	18.23806	17.959663	1.023845	0

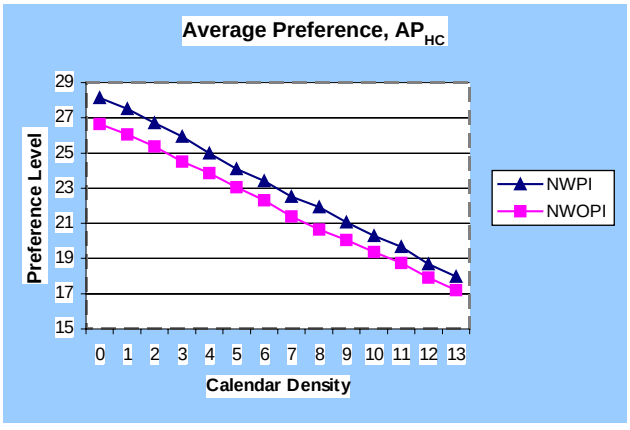

Chart 1. The average preference levels of each scheduled meeting.

decrease with the calendar densities. This chart also compares the gain in preference level obtained with Algorithm 2 using *preference estimation* compared with Algorithm 1 using only *relaxation*.

We also tested whether our algorithms have a bias towards any of the meeting participants. In this experiment, we arranged most of the meetings to be hosted by Agent 0. Charts 2 and 3 show the average deviation of preference levels from "expected" for each participant. Chart 2 shows that Algorithm 1 (NWOPI) has a strong bias for Agent 0 (A0)—the meeting initiator. Obviously, this is due to the fact that Algorithm 1 does not receive any preference level indications from the other participants and hence must rely solely on its own preferences in ranking and selecting alternatives. However, what is interesting is that Algorithm 2 (NWPI) produces very similar deviations for each agent; each agent is treated equally in terms of meeting their preference expectations. This is despite the fact that the whole negotiation process is managed by one MA that uses the initiator's preferences as initial "blue print" for the negotiation.

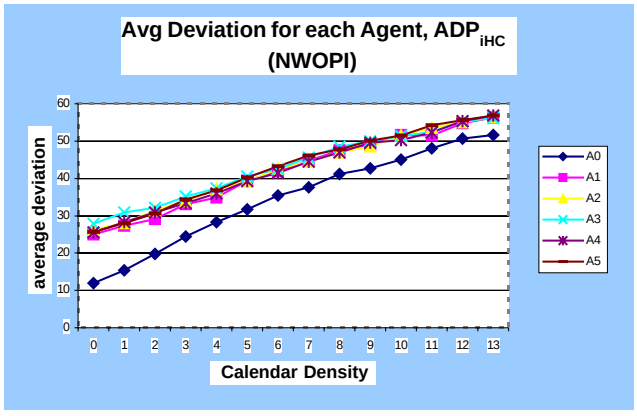

Chart 2. The average deviation for each agent using Algorithm 1 (NWOPI).

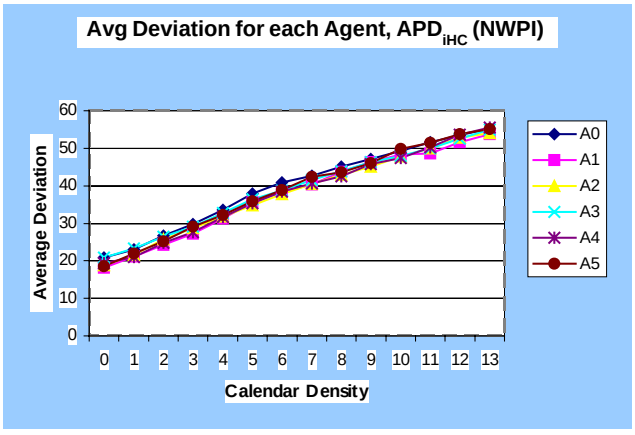

Chart 3. The average deviation for each agent using Algorithm 2 (NWPI).

Table 2
Success rates and communication cost

CD	Meeting success rate		Proposal success rate		Communication cost (Cycles)	
	Alg-1	*Alg*-2	*Alg*-1	*Alg*-2	*Alg*-1	*Alg*-2
0	0.9479	0.9544	0.6648	0.6572	2.5244	10.13
1	0.9248	0.9343	0.5949	0.5509	2.8119	10.092
2	0.8998	0.9059	0.5384	0.4738	3.1988	10.373
3	0.8739	0.8774	0.4905	0.4187	3.5494	10.726
4	0.8464	0.8415	0.4704	0.3878	3.8572	11.115
5	0.8126	0.8138	0.4365	0.3525	4.2626	11.634
6	0.7856	0.7854	0.4103	0.3234	4.6046	12.135
7	0.7547	0.7564	0.3838	0.2919	4.8802	12.52
8	0.7262	0.7337	0.3571	0.2798	5.1911	12.738
9	0.701	0.7062	0.3468	0.2651	5.2168	13.198
10	0.6816	0.6789	0.3255	0.2487	5.4258	13.235
11	0.6573	0.6583	0.3162	0.2343	5.5664	13.518
12	0.6277	0.6243	0.292	0.2164	5.7701	14.012
13	0.6044	0.6019	0.2813	0.2028	5.6918	14.08

We also computed the average success rates for meetings and proposals as well as the number of cycles required to schedule a meeting for different calendar densities (CD) (see Table 2). We found the on average, Algorithm 2 (NWPI) requires more negotiation cycles to complete compared to Algorithm 1 (NWOPI). In other words, Algorithm 1 finds a fast solution, which might not be optimal, while Algorithm 2 finds an optimal solution but requires a few more cycles. It is interesting to note that the meeting success rate for Algorithm 2 is actually slightly higher than Algorithm 1. In other words, finding optimal solutions actually increases the likelihood of success. The proposal success rate for Algorithm 2 is of course lower as more proposals need to be negotiated in order to find the optimal one. However, this difference is not substantial.

8. Conclusion

In this paper, we presented MAFOA's approach to distributed agent-based meeting scheduling. We also outlined a novel *preference estimation* technique that allows optimal meeting schedules to be found without complete knowledge of individual user preference models. Our approach to user preference modeling involves priorities, preferences and rules, and a method to use the user preference model in evaluating scheduling proposals through preference levels. We also described two meeting scheduling algorithms—Algorithm 1 (NWOPI) and Algorithm 2 (NWPI)—that are based on a negotiation protocol that uses this methodology. Using simulation, we verified that Algorithm 2, using *preference estimation*, finds the optimal solution at only a slightly higher cost that Algorithm 1, which relies on *relaxation*.

8.1. Conflict resolution extensions

Besides research on using *relaxation* and *preference estimation* techniques in negotiation, we have also performed research on *conflict resolution* as extensions to the above algorithms. Although full details of *conflict resolution* might be outside the scope of this paper, we would like to explain briefly how it helps improve our algorithms. As mentioned above, the attendee list may be further qualified with "must," "should" and "can." When a solution cannot be found in our previous Algorithms 1 and 2, the meeting is cancelled. With *conflict resolution*, the algorithm tries to resolve this *impasse* using a technique similar to *dependency directed backtracking* (Stallman and Sussman, 1977). In our case, the dependency is the source of the impasse—those previously committed events that conflict with the current event being scheduled. For each rejected proposal, the MA keeps track of *conflict levels* that serve as indications of the number of conflicts and the priorities of those conflicts. When an *impasse* occurs, the MA tries first to relax the problem by considering only "must" and "should" attendees. If no solution can be found, it then triggers rescheduling for previously committed but less important meetings. It does this by locating a previously rejected proposal where the number of conflicts is below a threshold and where the conflicting events are less important than the current events. This is done using a technique we call *conflict estimation*, which is similar to *preference estimation*. *Preference estimation* allows us to guess which proposal will most likely get accepted. *Conflict estimation* allows us to guess which proposal might cause the least amount of conflicts. If conflict resolution is successful, the current event gets committed. Otherwise, the meeting will be cancelled as before. The meeting will still be cancelled if the number of conflicts is above the threshold, i.e., too many meetings to reschedule and might not worth the trouble. This threshold can be dependent on the important of the current meeting being scheduled.

Ashir et al. (1997) introduces a concept called *quorum* and a *two-round negotiation protocol* that allow a meeting to be scheduled even when non-mandatory invitees are not available. However, in Ashir, *quorum* is used only as a way to relax the meeting-scheduling problem and to improve efficiency. In our case, the attendee list is used to trigger rescheduling if the current meeting cannot be scheduled because of conflicts with "must" and "should" attendees' schedule. Unscheduling (or moving) a lesser priority meeting to make time for another has also been studied by Woo et al. (1999). Their research is also based on agent negotiation using a contract net protocol (Sen and Durfee, 1996). However, Woo et al. does not use attendee qualifiers as we do to relax the problem prior to conflict resolution. Furthermore, rescheduling is performed by an agent that has complete knowledge of all the participants' schedule. In our case, the MA only knows *conflict levels* while details of individual's schedule remains hidden to protect privacy.

Acknowledgements

The work described in this paper was substantially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. 9040517, CityU 1109/00E). This work was also partially supported by a grant from City University of Hong Kong (Project No. 7001286).

References

Albrecht, K., Albrecht, S., 1993. Added Value Negotiating: The Breakthrough Method for Building Balanced Deals. Irwin Professional Publishing, Homewood, IL.

Ashir, A., Kwang, H.J., Kinoshita, T., Shiratori, N., 1997. Multi-agent based decision mechanism for distributed meeting scheduling system. In: Proceedings of the 1997 International Conference on Parallel and Distributed Systems, pp. 275–280.

Biswas, J., Bhonsle, S., Tan, C.W., Tay, S.Y., Wang, W., 1992. Distributed scheduling of meetings: a case study in prototyping distributed applications. In: Proceedings of the Second International Conference on Systems Integration, ICSI '92, pp. 656–665.

Bui, H.H., Venkatesh, S., Kieronska, D., 1995. A multi-agent incremental negotiation scheme for meetings scheduling. In: Proceedings of the Third Australian and New Zealand Conference on Intelligent Information Systems, ANZIIS-95, pp. 175–180.

Conry, S.E., Meyer, R.A., Lesser, V.R., 1998. Multistage negotiation in distributed planning. In: Bond, A.H., Gasser, L. (Eds.), Readings in Distributed Artificial Intelligence. Morgan Kaufman, San Mateo, pp. 367–384.

Danny, L., Mitsuru, O., 1998. Programming and Deploying Java Mobile Agents with Aglets. Addison-Wesley, Reading, MA.

Dawson, F., Stenerson, D., 1998. Internet Calendaring and Scheduling Core Object Specification (iCalendar). Proposed Standard, Network Working Group, RFC 2445, The Internet Engineering Task Force (IETF), November 1998.

Finin, T., Labrou, Y., Mayfield, J., 1997. KQML as an agent communication language. In: Brad-shaw, J.M. (Ed.), Software Agents. AAAI Press/The MIT Press, Cambridge, MA, pp. 291–316 (Chapter 14).

FIPA Interaction Protocol Library Specification, 2000. Foundation for Intelligent Physical Agents (FIPA).

Garrido, L., Sycara, K., 1996. Multi-agent meeting scheduling: preliminary experimental results. In: Proceedings of the Second International Conference in Multi-Agent Systems (ICMAS'96), Kyoto, Japan, December.

Guttman, R.H., Maes, P., 1998a. Agent-mediated integrative negotiation for retail electronic commerce. In: Proceedings of Workshop on Agent Mediated Electronic Trading, AMET-98.

Guttman, R.H., Maes, P., 1998b. Cooperative vs. competitive multi-agent negotiations in retail electronic commerce. In: Proceedings of the Second International Workshop on Cooperative Information Agents (CIA'98), Paris, France, July 3–8.

Higa, K., Shin, B., Sivakumar, V., 1996. Meeting scheduling: an experimental investigation. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol. 3, pp. 2023–2028.

Huhns, M.N., Stephens, L.M., 1999. Multiagent systems and societies of agents. In: Weiss, G. (Ed.), Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge, MA.

Jennings, N.R., Jackson, A.J., 1995. Agent-based meeting scheduling: a design and implementation. Electronics Letters 31 (5), 350–352.

Lange, D., Chang, D.T., 1996. IBM aglets workbench—programming mobile agents in Java. White Paper, IBM Corporation, Japan, August 1996.

Lawler, E.L., Wood, D.W., 1966. Branch and bound methods: a survey. Operations Research (ORSA) 14, 699–719.

Liu, J.S., Sycara, K.P., 1994. Distributed meeting scheduling. In: Proceedings of the 16th Annual Conference of the Cognitive Science Society, Atlanta, GA, August 13–16.

Luo, X., Leung, H.-F., Lee, J.H.-M., 2000. A multi-agent framework for meeting scheduling using fuzzy constraints. In: Proceedings of the Fourth International Conference on MultiAgent Systems, pp. 409–410.

Park, S., Birmingham, W.P., 1995. Multiagent Negotiation Framework. Technical Report (CSE-TR-248-95), University of Michigan, Ann Arbor.

Pino, J.A., Mora, H.A., 1997. Scheduling meetings with guests' approval. In: Proceedings of the XVII International Conference of the Chilean Computer Science Society, pp. 182–189.

Pino, J., Mora, A., Hugo, A., 1998. Scheduling meetings using participants' preferences. Information Technology and People 11 (2), 140–151.

Schwartz, R., Kraus, S., 1997. Negotiation on data allocation in multi-agent environments. Artificial Intelligence 94 (1–2), 79–98.

Sen, S., 1997. Developing an automated distributed meeting scheduler. IEEE Expert 12 (4), 41–45.

Sen, S., Durfee, E.H., 1994a. On the design of an adaptive meeting scheduler. In: Proceedings of the Tenth IEEE Conference on Artificial Intelligence for Application, San Antonio, TX, March, pp. 40–46.

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

Sen, S., Durfee, E.H., 1996b. A contracting model for flexible distributed scheduling. Annals of Operations Research 65, 195–222.

Sen, S., Durfee, E.H., 1998. A formal study of distributed meeting scheduling group decision and negotiation. Group Decision and Negotiation Support System 7, 265–289.

Sen, S., Haynes, T., Arora, N., 1997. Satisfying user preferences while negotiating meetings. International Journal of Human–Computer Studies 47, 407–427.

Shapiro, S.C. (Ed.), 1992. Encyclopedia of Artificial Intelligence. Wiley, New York.

Smith, R.G., 1980. The contract net protocol: high-level communication and control in a distributed problem solver. IEEE Transactions on Computers C-29 (12), 1104–1113.

Stallman, R.M., Sussman, G.J., 1977. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. Artificial Intelligence 9 (2), 135–196.

Tsuchiya, K., Takefuji, Y., Kurotani, K., Wakahara, K., 1996. A neural network parallel algorithm for meeting schedule problems. In: Proceedings of the 1996 IEEE TENCON'96, Digital Signal Processing Applications, Vol. 1, pp. 173–177.

Wong, Y.M., Ho, T.T., Fung, K.L., Chun, H.W., 2000. A model for resource negotiation using mobile agents. In: Proceedings of Fourth World Multiconference on Systemics, Cybernetics and Informatics (SCI 2000), Orlando, FL, July 23–26.

Woo, S.J., Jeong, S.Y., Geun, S.J., 1999. Cooperation in multi-agent system for meeting scheduling. In: Proceedings of the IEEE Region 10 Conference (TENCON 99), Vol. 2, pp. 832–835.

Yang, G.-H., 1995. Running Meetings Effectively (Ye Wu Hui Yi Liang Ce), Japanese original by Nihon Seisansei Honbu. Wan Li Shu Dian, Hong Kong, ISBN: 962-14-0968-3 (Chinese translation).