ELSEVIER

Advanced Engineering Informatics 17 (2003) 1–22

ADVANCED ENGINEERING INFORMATICS

www.elsevier.com/locate/aei

N*—an agent-based negotiation algorithm for dynamic scheduling and rescheduling

Hon Wai Chun*, Rebecca Y.M. Wong

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

Received 2 November 2001; accepted 31 October 2002

Abstract

This paper presents a generalized agent-based framework that uses negotiation to dynamically and optimally schedule events. Events can be created dynamically by any active agent in the environment. Each event may potentially require collaboration or resources from one or more other agents. The allocation of resources to the event will be negotiated iteratively until a compromise is found. The framework consists of a user preference model, an evaluation or utility function, and a negotiation protocol. The negotiation protocol is used to implement a distributed negotiation algorithm, called *Nstar* (N*). N* is based conceptually on the A* algorithm for optimal search but extended for distributed negotiation. N* is a general distributed negotiation algorithm that makes use of *negotiation strategies* to find solutions that meet different negotiation objectives. For example, it can use a utility optimizing strategy to find the solution that maximizes average utilities of individual agents. Alternatively, it can select a time optimizing strategy to locate a 'quick' feasible solution. The negotiation protocol also performs conflict resolution using a form of iterative repair that *renegotiates* events which have conflicts. A special case of this framework was used in our MAFOA (mobile agents for office automation) environment to perform agent-based meeting scheduling. A computer simulation test bed was built to simulate the scheduling of hundreds of randomly generated meetings using our N* algorithm.
© 2003 Elsevier Ltd. All rights reserved.

Keywords: Distributed scheduling; Agent-based negotiation; Distributed artificial intelligence; A*; Meeting scheduling

1. Introduction

Our N* negotiation algorithm is a type of distributed AI (DAI) algorithm [8,37]. It works within a distributed agent-based environment [4] where agents might not have knowledge of other agent's rules, constraints, parameters, and preferences. N* allows agents to negotiate an optimal solution despite this lack of complete knowledge of the environment.

If knowledge was complete, then a negotiation problem 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 scheduling [36]. However, in reality, constraints and preferences related to an agent might not be accessible to other agents. For example, in a distributed meeting scheduling problem, 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 preference.

Meeting scheduling is commonly modeled as a type of distributed scheduling [29] problem. Each person is a distributed scheduler that manages its own local resources. A meeting schedule is produced when all these distributed schedulers collaborate to find a solution that satisfies not only their individual constraints and preferences but also the global ones. For example, one form of collaboration is through a generalized contract net protocol [30,33].

In this paper, we model scheduling as an agent-based process of negotiation. The scheduling of an event is performed by a software agent that negotiates with other agents that need to participate or collaborate in that event. Each agent's prior committed schedule, preferences and constraints are hidden from all other agents. By insulating the negotiation process from details of the user model, we enable our N* negotiation algorithm to work in

* Corresponding author. Tel.: +852-2788-7194; fax: +852-2784-4242.
E-mail addresses: andy.chun@cityu.edu.hk (H.W. Chun), rebecca.wong@alumni.cityu.edu.hk (R.Y.M. Wong).

1474-0346/03/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S1474-0346(03)00019-3

heterogeneous environments where agents might be built from different agent technologies with different user models and preference strategies. Negotiation in a heterogeneous environment is performed through a well-defined negotiation protocol.

Creating a user preference model and designing an algorithm that can perform distributed scheduling without detailed knowledge of the individual preference models are the main objectives of our research. Our scheduling algorithm uses a technique called 'preference estimation' that allows us to 'estimate' the preference model. *Preference estimation* is similar to using *underestimations* in search evaluation functions combined with dynamic programming.

In this paper, we document the key components within our negotiation framework (see Fig. 1), the details of the N^* negotiation algorithm, interactions of the agents and the computer simulation.

The framework consists of six main components

- *User Preference Model*. Each Collaborating Agent has a separate User Preference Model that encapsulates priorities, preferences and rules related to an individual collaborating agent.
- *Utility Function*. The User Preference Model is used by the Utility Function in evaluating proposals and counter proposals to come up with a *preference level* that influences the negotiation process.
- *Initiating Agent*. This agent is the initiator of the event that needs to be negotiated. It also plays the role of a coordinator and manages the whole negotiation process. There is only one initiating agent per negotiation problem.
- *Collaborating Agents*. Besides the initiating agent, there may be one or more Collaborating Agents that represent the other participants of the event. The collaborating agent may represent a particular user or an organization.
- *Negotiation Protocol*. Agents communicate with each other during negotiation using the Negotiation Protocol
- *Negotiation Algorithm*. The N^* negotiation algorithm defines sequences of interaction between the agents to perform a distributed search for an optimal solution.

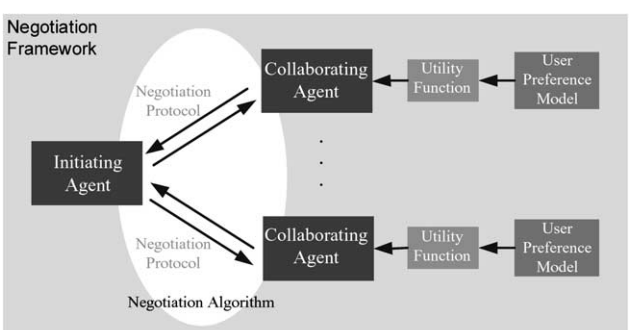

Fig. 1. Key components in our negotiation framework. Boxes represent software components, modules, or agents. Arrows indicate information or data flow.

The main purpose of the N^* negotiation algorithm is to find an optimal solution to a negotiation problem. The N^* distributed scheduling process is fully automated; scheduling responsibilities are fully delegated to the software agents. Other researchers [2] have investigated distributed 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 [1] 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 attention 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 [26], auctioning, and retail electronic commerce [12]. Different negotiation mechanisms [13,23] and strategies have been studied.

Huhns and Stephens [15] define agent-based negotiation as: "A frequent form of interaction that occurs among agents with different goals… 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". The major features of negotiation are the language used by the participants—the protocol that defines the legal actions during negotiation—and the decision process that the agents use. The decision process is basically the negotiation strategy that the agents use.

Sandip Sen and Edmund H. Durfee performed a formal study on distributed meeting scheduling [27] and studied the usefulness of different heuristic negotiation strategies to solve distributed meeting scheduling. They have identified different forms of conflicts between scheduling processes that adversely affect the scheduling efficiency of the system and presented different search heuristics. They stated that commitment strategies affect the frequency of conflicts between concurrently active scheduling processes. However, search biases affect the frequency of the conflicts between new processes and processes that have already finished scheduling their meeting requests [27,30]. The results from their analysis on different heuristic strategies were used to design an adaptive meeting scheduler. They have also demonstrated the benefits of adaptive choice of heuristic strategy combinations during meeting scheduling [31]. Although, Sen and Durfee did not take user preference into consideration during the scheduling process in Ref. [27], later in Ref. [28] a user preference mechanism based on a voting scheme was presented. They model preferences as elections between different alternative proposals.

Our user preference model is somewhat different. A formalized user preference model for a user is encapsulated within each of our agents. This model is personal and is only accessible by the associated agent. Although the model is encapsulated, the agents make use of the model to evaluate the 'goodness' of each proposal. The resulting goodness measure, which we call *preference level*, can be shared and is normalized for all agents. The N* algorithm presented in this paper makes use of the user preference model and preference levels to negotiate an optimal solution.

Other researchers [11,21] 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 but there is no guarantee that a schedule with the highest joint utility value will be found. User preference is also used in Refs. [11,21] where each agent specifies a set of meeting preferences for each meeting to be scheduled. Their model used three key attributes—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. We found that our model can be more precise in defining user preferences. In Refs. [11,21], it is assumed that the values closest to the preferred ones are better. However, this might not always be the case. For example, a user who prefers meetings on Monday and Friday might not necessarily prefer Tuesday or Wednesday.

2. User preference model

A key component of our negotiation framework is the formalization of a user preference model. Studies [14] show that fully automated agent-based 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 or an organization. Our agent might act on behalf of a person or an organizational entity, such as a particular department. The user preference model tries to encapsulate knowledge on different types of personal or organizational preferences and how they might change according to changes in the environment. The model influences the behavior of the software agents during negotiation. Its main purpose is to encode 'preferences' and not hard constraints or business rules. Constraint or rule engines may be used instead to encode this type of knowledge and supplement our preference model.

In our negotiation framework, each *negotiation problem* or *negotiation event* E_j has an importance level i_j that might depend, for example, on the event type or the rank of the agent that initiated the event. The importance level is used in conflict resolution (CR) to determine which events have lower importance or priority and might be candidates for renegotiation, i.e. rescheduling.

Each event E_j 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. For example, in a meeting-scheduling problem, the 'day' might be fixed to be on Tuesday, while the 'time' and 'location' attributes might be variables and negotiated. In addition, each variable attribute v_i has 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. For example, the value of 'time' might be '9 am', '10 am', 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 solution or 'compromise' 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 organization or 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. The evaluation results in a utility or a *preference level* for each proposal and counter proposal. The following sections provide detail definitions of *priorities*, *preferences*, and *rules* that constitute the user preference model. The preference model is encapsulated within an agent and is not accessible to others in the environment. However, agents may communicate their preferences of particular proposals or counter proposals but only to the initiating agent who acts as the negotiation coordinator.

2.1. 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}.

$$\sum_{x=1}^{n} ap_x = AP_{total} \quad (1)$$

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}. The 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 calculated as follows:

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

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

$$ap_x = AP_{total}/n \quad (3)$$

2.2. 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}.

$$\sum_{x=1}^{k} pv_x = PV_{total} \quad (4)$$

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 $pv_x = PV_{total}/k$ (Eq. (5)), where k is the total number of values in domain d_i.

2.3. Preference rules

Besides 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 our negotiation framework, 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 do not 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 the framework, 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.
maxAP( )—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.
minAP( )—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.
maxPV( )—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.
minPV( )—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.

3. Utility function

The user preference model provides knowledge on the preferences of an individual collaborating agent. This preference is measured in terms of a *preference level*. The *proposal evaluation function*, on the other hand, provides global preference information to the initiating agent to guide the N^* negotiation process. Besides using a *preference level* measure to evaluate acceptable proposals, we also defined a *conflict level* measure to evaluate unacceptable proposals. The *conflict level* indicates the degree that scheduled events conflict with the event being negotiated.

In our negotiation framework, a potential solution to a negotiation problem is defined as a tuple from $d_1x\ d_2x\ldots xd_n$ such that the n assignments of values to the problem's variable attributes is to the 'satisfaction' of all the collaborating or negotiating agents, i.e. a compromise is found. During the negotiation process, each agent will need to evaluate how 'satisfied' it might or might not be with the proposed solution or compromise, using the user preference model. In our framework, each proposed solution to a specified *negotiation problem* 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, i.e. a collaborating agent.

3.1. Preference level

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 the 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 am').

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

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

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.

Table 1
Calculation of the preference level for a meeting proposal #1

Meeting proposal #1	Agent #1		Agent #2	
	Priority	Preference	Priority	Preference
Part of the day: lunch	2.86	0	1.43	3
Day of the week: Monday	1.43	1	2.86	2
Total PL		1.43		10.01

A proposal/counter proposal with a higher preference level means it is more preferred by that user. To illustrate how the user preference model is used to evaluate proposals/counter proposals and to determine their preference levels, let us use a simplified example of a meeting negotiation problem with only two variable attributes: 'part of the day' and 'day of the week' (see Table 1).

The above proposal is to have the meeting during lunch on Monday. The priorities and preference values are different for different people. For agent #1, the part-of-the-day is more important than day-of-the-week. Its preference for having a meeting during lunch and meeting on Monday is very low. The resulting preference level for this proposal is only 1.43 as determined by Eq. (5), i.e. $2.86 \times 0 + 1.43 \times 1 = 1.43$. For agent #2, the priority is reversed, part-of-the-day is lower than day-of-the-week. It has a preference for having meeting during lunch and a slight preference for Monday. The resulting preference level becomes $10.01 = 1.43 \times 3 + 2.86 \times 2$. Therefore, if meeting proposal #1 was raised, agent #1 will most likely like to negotiate a better proposal, while agent #2 might find the proposal totally acceptable. Luo et al. [22] take an alternative approach and use fuzzy constraints to model preferences. We find our model adequate for the purpose of determining a *preference level* for a proposal. Pino et al. [24,25] offer a graphic user interfaces based on rules and timeslots to define meeting preferences, which they call *lateral model*. In their case, only a human meeting coordinator uses the preference information. For our model, only an agent has access to the user preference model.

3.2. Proposal evaluation function

The preference level only represents the preferences of a particular collaborating agent. The initiating agent that coordinates the negotiation process might need to evaluate the 'global' preference of a proposal to determine its negotiation strategy and which proposals to make. In our

framework, this global preference is calculated using a *proposal evaluation function*.

Our *proposal evaluation function* is similar to the role of the evaluation function in A* [35] that estimates the total cost of a solution. The evaluation function $f'(n)$ in A* is given by:

$$f'(n) = g(n) + h'(n) \tag{6}$$

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 negotiation algorithm, n is analogous to a particular negotiation cycle. The actual costs are the actual preference levels of the proposals or counter proposals received by the initiating agent from the collaborating agents. The estimated costs are the 'estimated' preference levels of new proposals that have not yet been reviewed by the collaborating agents. In our N* algorithm, we use the 'current' or 'minimum' preference level as the *estimated preference level*. This approach guarantees that optimal solutions will not be 'overlooked'; since the actual preference level will in fact be lower. This usage of the minimum value is similar to 'underestimation' in A* search algorithms.

Only the initiating agent needs to calculate the *proposal evaluation function* and the *estimated preference levels*. For the initiating agent, the estimated preference level $\mathrm{pl}'_i(P_x)$ of any proposal/counter proposal P_x of collaborating agent i is defined as:

$$\mathrm{pl}'_i(P_x) = \begin{cases} \mathrm{pl}_i(P_x), & \text{if } P_x \text{ has already been proposed/counter proposed} \\ \min(\mathrm{pl}_i(P_x)|_x) & \text{for all } P_x \text{ proposed/counter proposed} \end{cases} \tag{7}$$

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. Our N* algorithm requires the actual preference level to be passed back from the collaborating agents to the initiating agent when they counter propose. Otherwise, we will use the minimum preference level received so far from the individual collaborating agent. This approach is equivalent to the 'most recent' preference level, as the value will be decreasing—collaborating agents will always propose the 'best' or 'most preferred' alternatives first.

Instead of adding the costs, as in A*, our N* proposal evaluation function computes the *average preference level*. With the ability to calculate the $\mathrm{pl}'_i(P_x)$ value for every participant and every proposal, the initiating agent can then compute the proposal evaluation function $f(P_x)$, i.e. equivalent to global *average preference level* $\mathrm{apl}(P_x)$, for each proposal P_x :

$$f(P_x) = \mathrm{apl}(P_x) = \frac{\sum_{i=1}^{n} \mathrm{pl}'_i(P_x)}{n} \tag{8}$$

where n is the total number of participants in the event.

The *average preference level* provides a global indication of how preferred one proposal might be compared with another and hence indicates 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.

Note that the preference levels from the individual collaborating agents are only sent to the initiating agent. No other agent has knowledge of these preference levels neither does it have access to individual preference models. Our use of an *average preference level* is similar to Jennings and Jackson [16] where an average preference level is computed from the individual preference levels that are passed back. However, [16] does not use *preference estimation* to rank the remaining proposals. 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 agent.

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/collaborate in the event, a group that 'should' participate, and a group that 'can' participate if they wish. For these cases, N* uses a weighted evaluation function to give more consideration to preferences from the must and should group:

$$f(P_x) = \frac{\sum_{i}^{\text{must}} w_m \mathrm{pl}'_i(P_x) + \sum_{j}^{\text{should}} w_s \mathrm{pl}'_j(P_x) + \sum_{k}^{\text{can}} w_c \mathrm{pl}'_k(P_x)}{n} \tag{9}$$

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,
w_c	is a weight factor for the can group and is usually less than or equal to 1.0.

3.3. Conflict level

The preference level allows N* to determine which proposals are better than others. Besides this measure,

collaborating agents will also compute a *conflict level* (*cl*) for rejected proposals that is used as justification for the rejection. The conflict level indicates the degree of conflict a proposal has with previously scheduled events. In our research, we assume agents are cooperative (i.e. an agent will accept a proposal if the user is available, even though that proposal might be of lower preference) and that a proposal is rejected only if necessary resources are used by other scheduled events. The conflict level is proportional to the number of conflicts a proposal has with previously scheduled events, the priority of those events and the level of participation (must participate, should participate or can participate). The conflict level is evaluated using Eq. (10):

$$\mathrm{cl}(P_k) = \sum_{i}^{n} p_i w(l_i) \tag{10}$$

where

n	is the existing event conflicting with P_k
p	is the priority/importance of the conflicting event n
$w(l_i)$	is a weighting factor for the level of participation, l_i, of the conflicting event n

Whenever an initiating agent cannot find a common agreeable schedule among all the must and should participants from all the possible proposals, the negotiation will enter what we call a CR stage. During CR, the initiating agent will request agents with conflicts to try to cancel or reschedule previously scheduled events. The collaborating agent will compare the importance of the event being negotiated and its *conflict level* with existing events to determine whether or not to cancel or reschedule those scheduled events. The importance of the event being negotiated is also calculated using Eq. (10), where n is the event being negotiated.

3.4. Conflict evaluation

The conflict level represents the degree of conflict a proposal has with existing events. Collaborating agents use the conflict level to determine whether or not to cancel or reschedule conflicting events. On the other hand, the initiating agent coordinating the negotiation will also need to determine which rejected proposal is most likely to get resolved among all rejected proposals. CR process will begin only when all possible proposals are rejected by some or all of the must or should participants. Using the *conflict level* sent back by collaborating agents as justification for rejecting proposals, the initiating agent can determine which rejected proposal causes the minimal amount of conflicts. The initiating agent can then select less conflicting proposals to use during CR. As with *preference estimation*, the initiating agent will perform *conflict estimation* (CE) during the CR process. The initiating agent will maintain a vector that stores the *estimated conflict* for each participant. The *estimated conflict* estCl_j, for *agent* j, is defined as:

$$\mathrm{estCl}_j = \begin{array}{c}\text{the lowest conflict level, cl, among those}\\ \text{proposal rejected by agent, } j, \text{ in CR.}\end{array} \tag{11}$$

By using *estimated conflict*, the initiating agent is able to decide whether rejected proposals that have not yet been processed by CR might still be viable alternatives. For example, if during CR an agent refuses to resolve conflicts of proposal P_k, which has conflict level equal to $\mathrm{cl}(P_k)$, it is most likely that the agent will not accept another proposal P_j with conflict level $\mathrm{cl}(P_j) \geq \mathrm{cl}(P_k)$. This allows the initiating agent to determine when to terminate the CR process. The CE is updated during each round of CR negotiation.

4. Agent structure

Agents in our negotiation framework can either play the role of an *initiating agent* or a *collaborating agent*. An initiating agent is the agent that initiated the negotiation problem, i.e., the event to be negotiated. Any agent in the environment can be the initiator of an event that needs to be scheduled or negotiated. All the other participants of an event play the role of the collaborating agent. There is only one initiating agent per negotiated event with one or more collaborating agents. Fig. 2 highlights the key agent structure (using a UML class diagram).

4.1. Collaborating agent

Each collaborating agent knows the committed calendar or schedule of the individual or organization it represents. The schedule is stored in a 'Schedule' data structure that is basically a timetable of events that have been committed or events that are being considered. The preferences are stored using the 'User Preference Model' defined earlier that consists of priorities, preferences and rules.

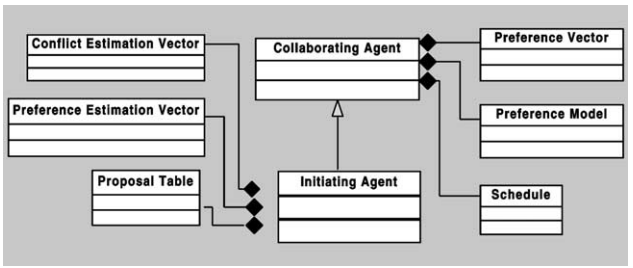

Fig. 2. UML class diagram of the Initiating and Collaborating Agent. The initiating agent is just like the other agents, except it keeps track of preference and CEs to improve the quality of the search, which is guided by the Proposal Table.

Given the schedule and preferences, we will then be able to determine when an organizational entity or person is available and what the preferences are for these available resources. For each *announced* event that the user is invited to participate, the collaborating agent will create a sorted 'Preference Vector (PV)' to store pre-computed preferences to be used during negotiation in evaluating proposals and in generating counter proposals.

4.2. Preference vector

The PV is a sorted vector of a person's preferences on different alternative proposals for an announced event, such as a meeting to be scheduled. The vector is sorted according to the value of the *preference level* of these alternatives. The preference level is computed based on Eq. (5). 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. In our negotiation framework, only proposals that the user is willing to commit to are included, i.e., proposals without any conflicts.

The PV contains a 'current' pointer (Fig. 3) that points to the current counter proposal that has been offered by this agent to the initiating agent. Each agent 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 agent has made.

4.3. Initiating agent

The main role of an initiating agent is to coordinate the negotiation or scheduling process of an individual event. It also represents the interest of the user that initiated the event when negotiating with other agents. When an agent plays the role of an initiating agent, it has additional data structures—the *proposal table*, the *preference estimation vector* (PEV) and the *conflict estimation vector* (CEV).

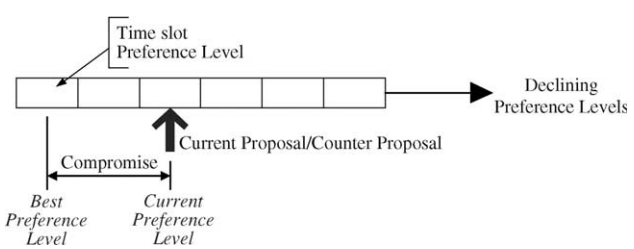

Fig. 3. The structure of the PV. This is a data structure that each agent uses to decide which proposal to accept or decline, and which proposals to counter offer.

4.4. Proposal table

The initiating agent uses the PEV to generate proposals to offer to the collaborating agents participating in the event being negotiated. In return, each collaborating agent can firstly accept or refuse the proposal (with a reason), and secondly offer a counter proposal from its PV. These proposals and counter proposals are all stored and tallied in a 'Proposal Table' data structure that allows the initiating agent to make negotiation decisions.

The proposal table has an entry for each proposal and counter proposal together with information on which agent accepts or rejects those proposals together with the actual preference level for acceptable proposal, which will be used by the N^* evaluation function. If a proposal was rejected, the corresponding conflict level will also be stored in the table. This information will be used during CR in case a compromise cannot be found.

The proposal table provides additional information to an initiating agent 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 counter proposals that have been offered by a large number of collaborating agents. Or, it may decide to go for a high quality solution by continuing the negotiation until the optimal solution that maximizes its *proposal evaluation function* is found.

4.5. Preference estimation vector

Preference estimation is a technique that allows an initiating agent 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. Some researchers use negotiation algorithms that keep preferences or utility information totally private, i.e. a *private preference* model as defined in Ref. [11]. Others use a *public preference* model [11]. Our framework is midway; no agent has access to any other agent's preference model. However, the initiating agents will know of the preference levels (not models) of other collaborating agents.

The PEV is used to perform preference estimation. It is similar to the PV (see Fig. 3) 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 agent, the PEV consolidates actual and 'estimated' preference levels (as defined in Eq. (7)) from all the collaborating agents and is dynamically sorted according to the N^* evaluation function, i.e. the proposal evaluation function (Eq. (8)). Only the initiating agent has the PEV. The initiating agent drives the negotiation process by selecting the next n best proposals from the PEV.

4.6. Conflict estimation vector

The CEV is a technique that allows an initiating agent to determine which conflicting proposal has a higher chance of being resolved and also be able to terminate the CR process earlier if there is no possible hope of resolution. Each initiating agent maintains a CEV. Entries in CEV store the *estimated conflict* for each of the participants. The estimated conflict is the lowest *conflict level*, cl(P), among all rejected proposals that have been reviewed during the CR stage. The estimated conflict acts as an upper bound; any proposal with conflict level greater than that can be ignored.

All previously rejected proposals will be sorted using the largest conflict level returned by the collaborating agents. The initiating agent drives the CR process by selecting the proposals with least amount of conflicts and within the conflict level upper bound.

5. The negotiation protocol

The negotiation protocol defines the interactions between agents during negotiation and also the rules and assumptions of the negotiation process. Our negotiation protocol is a form of multistage negotiation protocol [7] and is similar to the protocol used in Refs. [16,27,30,31,32,28]. The content of the negotiation messages can be in a standard knowledge representation, such as KQML [9]. Fig. 4 lists the negotiation protocol used in our negotiation framework.

The following are potential messages that might be sent from the initiating agent to the collaborating agents. All these messages are announcements that are broadcasted or multicasted asynchronously to all the potential participants of an event.

Announce Event. Details of an event, such as the fixed and variable attributes of the event. For example, in meeting scheduling, an event announcement might be: 'Tom would like to hold a departmental meeting next week'. In addition, for variable attributes, the initiating agent may further limit the domain of the variable attributes to fall within a given range or set of values.

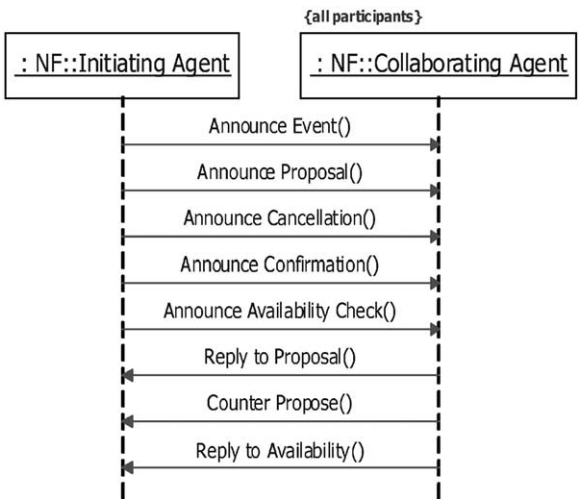

Fig. 4. UML Sequence Diagram listing the negotiation protocol. The negotiation protocol was intentionally designed to be simple.

Announce Proposal. A set of proposed values for the variable attributes of an event. For example, in meeting scheduling, a proposal announcement might be: 'How about next Monday morning?' The proposals are taken from the *preference estimation vector*, which is a vector of proposals sorted by the N^* evaluation function. With each negotiation cycle, the next n best proposal will be announced.

Announce Cancellation. This message tells all the collaborating agents that a particular negotiation event has been cancelled, either due to an impasse in negotiation or by another higher priority event during CR.

Announce Confirmation. Once a feasible solution has been found, this solution will be announced to all those involved in the event and a commitment will be made.

Announce Availability Check. Ask agents to check whether they would still be available if conflicting events, if any, were canceled or rescheduled. This message is only used during CR.

The following are potential messages that might be sent by the collaborating agents to the initiating agent.

Reply to Proposal. Either accept or reject a proposal. For acceptance, the actual preference level will be included in the reply. For rejection, the corresponding conflict level will be included. 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. A proposal is 'not acceptable' only when an agent has a prior committed event that overlaps/conflicts with the proposal. The conflict level will be used in the Cr algorithm to guide the resolution process.

Counter Propose. After receiving proposals, a collaborating agent can also counter propose. We assume that an agent will always counter propose with its next m best alternatives. Furthermore, we also assume that an agent will not counter propose if the original proposals from the initiating agent are already good enough, i.e. the preference level of a counter proposal must be higher than that of an original proposal. It is not logical to counter propose with a worse alternative.

Reply to Availability. This is similar to 'reply to proposal' except that no preference or conflict level will be returned. This message is only used during CR.

6. The N^* Algorithm

Our N^* negotiation algorithm makes use of the negotiation protocol to perform a distributed negotiated search for an optimal solution/compromise. Our N^* algorithm is based on the A^* optimal search algorithm. To compare N^* with A^*, Fig. 5 is the pseudo-code for the A^* algorithm.

```
Line
   1 PROCEDURE Astar (start, goal)
   2 BEGIN
   3   open←{start};                                  // nodes to be expanded
   4   closed←{};                                     // nodes already expanded
   5   solution←NIL;
   6   WHILE (open≠{})
   7   DO BEGIN
   8     node←RemoveFirst(open);                      // best node first
   9     IF (goal∈node & Cost(node)<Cost(solution))   // found potential solution
  10     THEN BEGIN
  11       solution←node;
  12       IF (Min(Cost(open))>=Cost(solution))       // branch-and-bound
  13       THEN RETURN (solution)                     // return optimal solution
  14     END
  15     ELSE BEGIN                                   // else continue search
  16       children←Expand(node);
  17       closed←Insert(node, closed);
  18       children←RemoveLoop(children);             // ignore paths with loops
  19       children←RemoveRedundant(open, children);  // ignore those already in open
  20       children←RemoveRedundant(closed, children); // ignore those that failed before
  21       open(AddFront(open, children);             // update open list
  22       open(Sort(open);                           // sort using underestimated cost
  23     END
  24   END
  25   RETURN (NIL);                                  // no solution
  26 END
```

Fig. 5. Pseudo-code for A* search algorithm.

A* performs a best-first search with branch-and-bound using an evaluation function that returns an underestimated total cost. Our N*, on the other hand, performs a best-first negotiation (offering the next best proposal with each negotiation cycle) with branch-and-bound using a *proposal evaluation function* that is also represents an underestimated total cost. Fig. 6 is the pseudo-code for our N* negotiation algorithm.

N* tries to propose solutions to the negotiation problem from a sorted list of potential proposals (Fig. 6—Line 1). The list is sorted according to the *proposal evaluation function* that performs 'preference estimation'. With each negotiation cycle (Fig. 6—Line 6), the next best proposal (Fig. 6—Line 8) will be offered/announced to all the collaborating agents (Fig. 6—Line 9). The definition of best may be different for different problem. The N* algorithm

```
Line
   1 PROCEDURE Nstar (proposals)
   2 BEGIN
   3   open←proposals;                                // proposals to be negotiated
   4   closed←{};                                     // proposals already negotiated
   5   solution←NIL;
   6   WHILE (open≠{})
   7   DO BEGIN
   8     node←RemoveFirst(open);                      // best proposal first
   9     AnnounceProposal*(node);                     // announce proposal
  10     replies←GetReplies*(node);                   // accept/reject proposals
  11     IF (Okay(replies) & Cost(node)<Cost(solution))  // found potential solution
  12     THEN BEGIN
  13       solution←node;
  14       IF (Min(Cost(open))>=Cost(solution))       // branch-and-bound
  15       THEN RETURN (solution)                     // return optimal solution
  16     END
  17     ELSE BEGIN                                   // else continue search
  18       children←GetCounterProposals*(node);       // consolidate counter proposals
  19       closed←Insert(node, closed);
  20       IF (∃x replies[x]={} & children[x]={})     // no reply & no counter proposal
  21       THEN RETURN ConflictResolution(closed);    // try conflict resolution
  22       children←RemoveInfeasible(children);       // ignore infeasible counter proposals
  23       children←RemoveRedundant(closed, children); // ignore those that failed before
  24       open←UpdateCost(open, children);           // update open w/ new preference est.
  25       open←Sort(open);                           // sort using underestimated cost
  26     END
  27   END
  28   RETURN (NIL);                                  // no solution
  29 END
```

Fig. 6. Pseudo-code for N* search algorithm.

allows different negotiation strategies to be implemented simply by redefining best. For example, a strategy to 'minimize negotiation time' can be implemented by defining best as the proposal with the maximal number of consenting counter proposals.

Collaborating agents can either accept or reject the proposal in their replies (Fig. 6—Line 10). If all collaborating agents accept the proposal and the cost (the reverse value of the *proposal evaluation function*) is lower than that of the previous potential solution (Fig. 6—Line 11), then that proposal becomes the new solution candidate. If no other alternative has a lower cost than the proposal, then it will be returned as the final solution (Fig. 6—Line 15). Because of branch-and-bound [20] the N* negotiation does not stop, even after a feasible solution has been found, until all other alternatives have lower evaluation function scores.

Otherwise, the counter proposals from all the collaborating agents will be collected and 'preference estimations' will be updated (Fig. 6—Line 18). A collaborating agent that rejects all proposals and has no counter offer means that agent has exhausted all its choices and there is no solution (Fig. 6—Line 20). The CR algorithm will be called to resolve this impasse.

Initially, differences between agents may be great. A proposal that is good for one agent might be bad for another. The N* negotiation process gradually reduces this difference until a common ground or compromise is found. Bui, et al. [5] 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 our negotiation framework, each proposal contains a complete set of proposed variable attribute values and the whole set is negotiated all at once, which seems more realistic and similar to real world scheduling.

7. The N* negotiation process

Since the N* negotiation algorithm is implemented as interactions between agents, we use a UML activity diagram (Fig. 7) to help illustrate the negotiation process. The interactions are similar to the negotiation process defined in

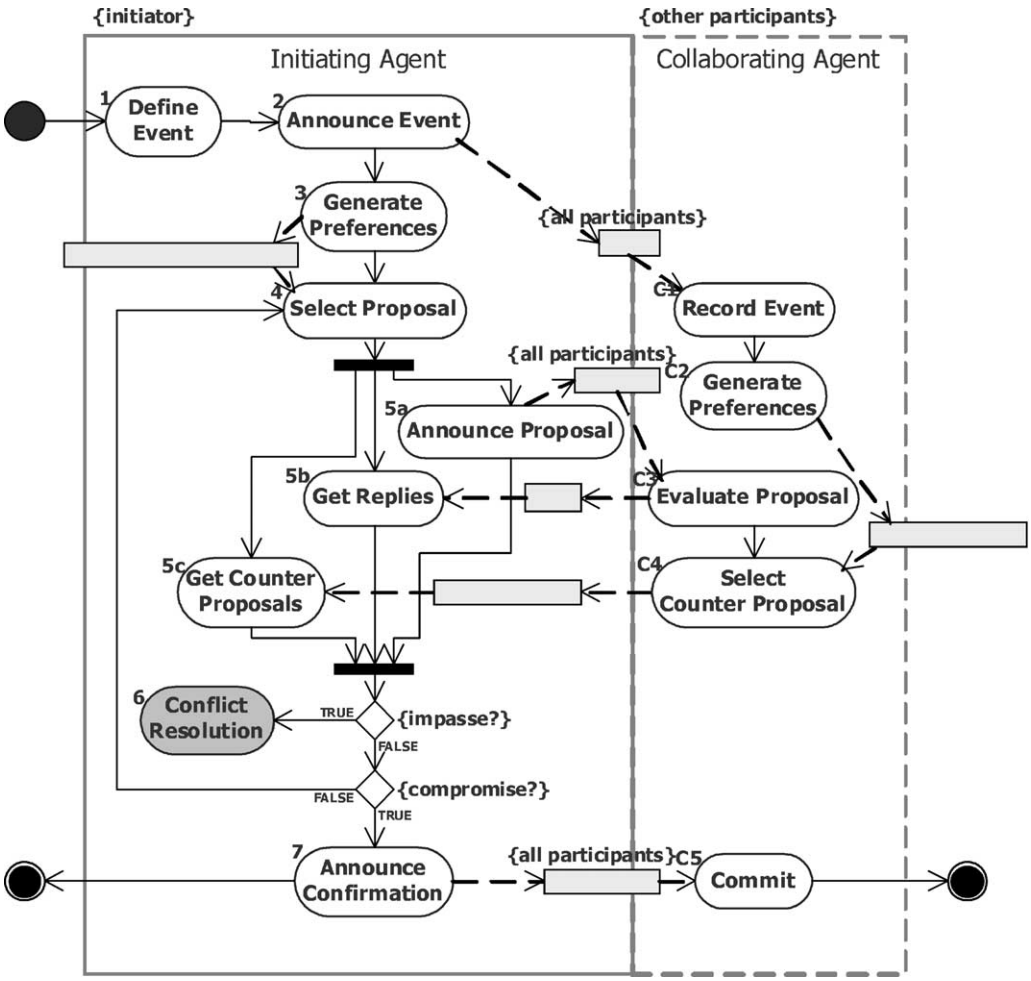

Fig. 7. UML Activity Diagram of the negotiation algorithm.

Ref. [27], but instead of simply using the earliest timeslot to guide the negotiation, which might not be the most preferred, we use our user preference model and N* evaluation function.

7.1. Negotiation activities

The activity diagram in Fig. 7 represents one initiating agent negotiating with one or more collaborating agents. The key action states in our algorithms are (the Step numbers refer to numbers in Fig. 7):

- *Step 1*. Define Event—the user initiating the negotiation problem provides details of the event to be scheduled to the initiating agent. The initiating agent will generate a list of possible proposals, which is based on the initiator's calendar and user preference model.
- *Step 2*. Announce Event—event details are then announced to all the collaborating agents, which include all the *fixed attributes* and their values, and the domain range of the *variable attributes*.
  - *Step C1*. Record Event—each collaborating agent records the new event and prepares to negotiate.
  - *Step C2*. Generate Preferences—each collaborating agent considers its calendar and preference model to produce a list of possible counter proposals, which are then stored in the *preference vector* and sorted according to their *preference levels*. Counter proposals are selected from this sorted PV.
- *Step 3*. Generate PEV—the *proposals* obtained from users (both the initiator and other collaborating agents) will be used together with *preference estimations* to form the PEV. The PEV will be updated every time when new information from participants is collected.
- *Step 4*. Select Proposal—the next n best proposals from the PEV will be selected. The PEV is dynamically sorted each time using the proposal evaluation function after consolidating preference levels received from collaborating agents during Step 5. Since the negotiation process will be directly driven from the PEV, this form of dynamic programming allows the initiating agent to dynamically change negotiation direction as more information is received through proposal acceptance messages or counter proposals.
- *Steps 5*
  - *Step 5a*. Announce Proposal—the selected proposals will be announced to all the collaborating agents.
    - *Step C3*. Evaluate Proposal—each collaborating agent evaluates each of the received proposals and returns a reply of either accept (with actual preference level) or reject (with actual conflict level for justification).
    - *Step C4*. Select Counter Proposals—each collaborating agent tries to select m counter proposals that have higher preference levels than those proposed by the initiating agent.
  - *Step 5b*. Get Replies—the proposal replies are consolidated and stored in the *proposal table*. Preference estimations are updated.
  - *Step 5c*. Get Counter Proposals—the counter proposals are consolidated and also stored in the *proposal table*. Preference estimations are updated.
- End Conditions:
  - *Case 1*: Impasse?—impasse occurs when any one agent rejects all proposals and has no counter proposal to offer. In this case, there is no obvious solution and CR will be performed. In N*, an impasse caused by participants who are not in the must and should participation groups can be ignored.
    - *Step 6*. CR—try to find a solution for the current negotiation problem by rescheduling or renegotiating lower priority events and hence eliminate conflicts that caused the impasse. This process will be explained in detail in the following section.
  - *Case 2*: Compromise?—if a consensus has been made and the optimal solution[1] is found, then the negotiation problem is solved and Step 7 will be executed.
    - *Step 7*. Announce Confirmation—an announcement is made to all agents that a proposal has been mutually accepted.
    - *Step C5*. Commit—agents commit the final solution into their calendars.
  - *Case 3*: No Compromise?—if none of the proposals are accepted in this negotiation cycle, the initiating agent must iterate another cycle and offer another set of proposals, i.e. go to Step 4.

7.2. Conflict resolution

The CR mechanism is engaged whenever there is an impasse. Our CR is similar to *dependency directed backtracking* [34]. In N*, the dependency is the source of the impasse—those previously committed events that conflict with the event being scheduled. For each rejected proposal, P_k, the initiating agent can find the *greatest conflict level*, $\max(\mathrm{cl}_j(P_k)|j)$, and the 'number of conflicts' (see Table 2). The CR mechanism finds and sorts previously rejected proposals first by the greatest conflict level in increasing order and within greatest conflict level in increasing order by the number of conflicts. Proposals for CR are then selected from this list.

Conflicting events that are lower priority than the current event being scheduled will be de-scheduled and renegotiated to free up time for the current event. Woo et al. [17] also studied canceling or rescheduling a lesser priority meeting to make time for another. Their research is based on agent negotiation using a contract net protocol [30].

In N*, each event E_j is associated with an *importance level* i_j that is a function of the attributes of the event. Using

[1] An optimal solution is a common interval, which has the best average preference level.

Table 2
A rejected proposal with number of conflicts equal to 3, and greatest conflict level, $\max(\mathrm{cl}_j(P_k)|j)$, equal to 30

	CA #1	CA #2	CA #3	CA #4
Proposal, P_k	Accept, $\mathrm{pl}_1(P_k) = 15$	Reject, $\mathrm{cl}_2(P_k) = 10$	Reject, $\mathrm{cl}_3(P_k) = 20$	Reject, $\mathrm{cl}_4(P_k) = 30$

Eq. (10), collaborating agents can calculate a conflict level to represent the importance of the event being scheduled. Collaborating agents will only try to cancel or de-schedule conflicting events of lesser importance than the current event E_j. The initiating agent will also make use of the conflict level received from the collaborating agent to determine whether the rejected proposal will be considered a candidate during CR.

A proposal, P_k may be considered a candidate for CR if it satisfies all the following conditions:

1. number of conflicts $\leq N_{\mathrm{conflict}}$, and
2. $\mathrm{cl}_j(P_k) < \mathrm{estCL}_j$, for all agent j, which reject P_k

where

N_{conflict} is a threshold that defines the maximum number of conflicts of a proposal,
$\mathrm{cl}_j(P_k)$ is the conflict level for the collaborating agent j with respect to proposal P_k,
estCL_j is the lowest conflict level of those proposals, which have been rejected by the collaborating agent j in the CR process. Initially, the estimation will be set to a large infinite value.

If a candidate solution or proposal requires a large number of conflicts to be resolved before it can be scheduled, then at a certain point it might not be worth the trouble resolving. The N_{conflict} threshold defines this point. Also during CR, if the collaborating agent refuses to cancel events that conflict with proposal P_k with a conflict level equal to cl_k, then it is likely that agent will also not accept other proposals P_j with conflict levels $\mathrm{cl}_j \geq \mathrm{cl}_k$. With CE, the initiating agent is able to determine when the CR algorithm should terminate without a solution. The CEV is updated during each round of CR negotiation.

The CR mechanism keeps a sorted list of candidate proposals to resolve. The list is sorted by the greatest conflict level, $\max(\mathrm{cl}_j(P_x)|j)$, of each rejected proposal and the number of conflicts. This list is called the *least conflict set*. The number of conflicts caused by each proposal in this list must be less than or equal to N_{conflict}. Thus, a least conflict proposal will be the one with the smallest greatest conflict level and the smallest number of conflict.

Fig. 8 shows the key action states in our CR algorithm (the Step numbers refer to numbers in Fig. 8; the '*' after the Step number indicates that the step belongs to the CR algorithm and differentiates it from step numbers in the initial negotiation algorithm shown in Fig. 7):

- *Step 1* *. Find Least Conflict Set—the *least conflict set* is computed and sorted. The next proposal with the least amount of conflict is then selected. The proposal P_k must have a number of conflicts less than or equal to N_{conflict} and the conflict of the individual collaborating agent j with respect to proposal P_k must be less than the estimated one (i.e. $\mathrm{cl}_j(P_k) < \mathrm{estCL}_j$).
- *Step 2* *. Announce Cancellation—the event being scheduled will be cancelled if the least conflict set is empty (i.e. no candidate proposals satisfy all the conditions, (1) and (2) defined above), or if all the proposals in the set have been tried and failed. Possibly the user may then decide to initiate a new event with modified event attributes.
  - *Step C1* *. Cancel—each collaborating agent will mark this event as cancelled.
- *Step 3* *.
  - *Step 3a* *. Announce Availability Check—check if the collaborating agents are still willing to accept the proposals if the conflicts are resolved. Other events may have been scheduled since the point in time when the agents first reviewed the proposal, and other environmental changes may affect the proposal acceptance.
    - *Step C2* *. Check Availability—each collaborating agent checks the proposal for acceptance by comparing the importance of the conflicting events and the meeting being negotiated.
  - *Step 3b* *. Get Replies—replies from the collaborating agents are consolidated and estimated conflicts are updated.
- End Conditions:
  - *Case 1*: Still in Conflict?—if not all collaborating agents accept the proposal, the next proposal with least conflicts will be tried (i.e. go to Step 1).
  - *Case 2*: Resolved?—if the proposal is accepted, the event will be confirmed and any conflicts will be renegotiated.
    - Step 4*. Announce Cancellation (Conflict)—all previously scheduled events that conflict with the proposal will be cancelled.
      - *Step C3* *. Cancel Conflict—each agent cancels the conflict from its calendar of committed events and notifies the coordinator of the cancelled event in order to withdraw its participation. Then the coordinator of the cancelled event will decide whether the event needs to be cancelled or rescheduled and will notify other related parties.
    - *Step 5* *. Announce Confirmation—an announcement is made to all the collaborating agents that a proposal has been mutually accepted.
      - *Step C4* *. Commit—each agent commits the final solution in its calendar of events.
    - Step 6*. Renegotiate Conflict Event—the coordinator of the cancelled event takes responsibility to

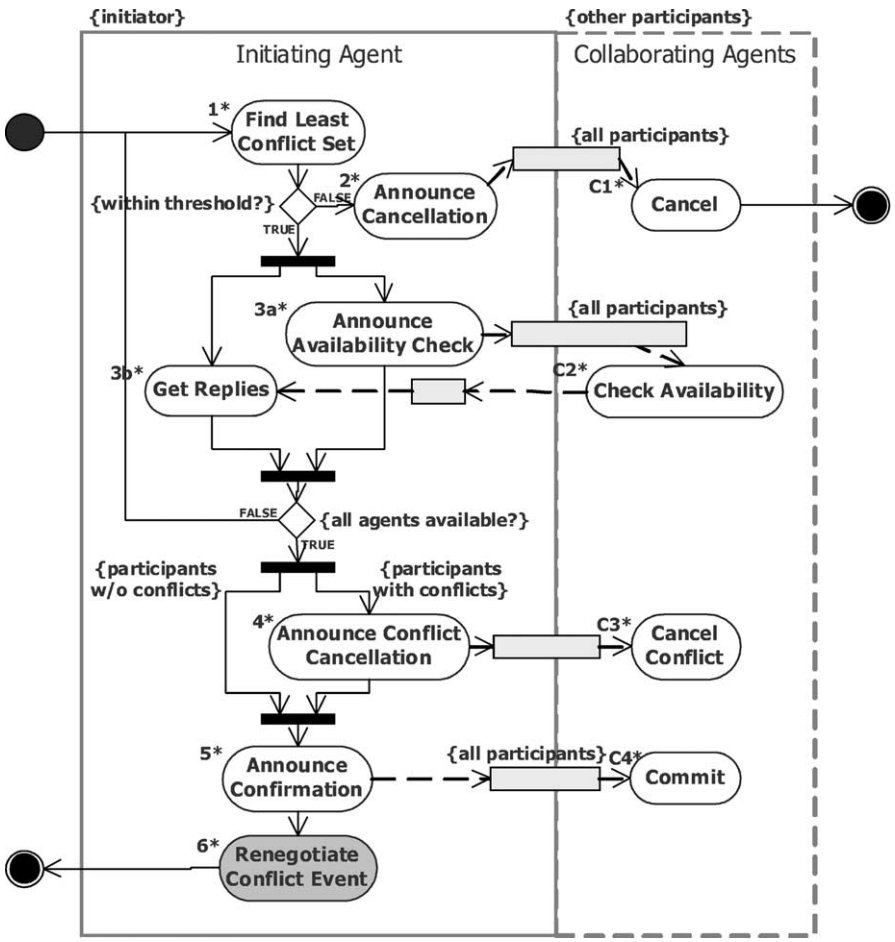

Fig. 8. UML Activity Diagram of the CR algorithm.

renegotiate and reschedules using the same negotiation process as documented in Fig. 7.

8. Computer simulations

8.1. Simulation parameters

We have applied the N* negotiation algorithm to the classic meeting scheduling problem and developed a computer simulation test-bed to simulate the scheduling and negotiation processes involved in scheduling different types of meetings in a distributed meeting scheduling problem. In our simulation program, we were able to specify a set of simulation parameters:

- Number of simulation cycles—the total number of times the whole simulation is to be executed
- Hours of meetings (H)—the total number of hours of meetings to be scheduled in a simulation cycle
- Max number of participants—the 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, it may be desirable to restrict the scheduling of meetings to regular office hours.
- Calendar density (CD)—the number of hours that are already preoccupied by prior engagements. The use of CD simulates the effect of having some timeslots pre-allocated from previous meeting scheduling exercises or for other non-meeting events such as annual leave, training, holidays, etc.
- Number of proposals (n)—the number of proposals sent from the initiating agent to each collaborating agent during each cycle
- Number of counter proposals (m)—the number of counter proposals to be returned from each collaborating agent to the initiating agent during each negotiation cycle

The initial state of each attendee's calendar is determined by the parameter CD, which indicates the number of occupied slot in the attendee's calendar. The exact slots that are to be occupied are generated randomly, i.e. if CD = 5, the program will randomly generate five 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

Table 3
Meeting length distribution

Length: L	1	2	3	4	5	6
Distribution: Pb (L)	0.25	0.3	0.25	0.1	0.05	0.05

simulation purposes, meeting lengths are randomly generated according to the distribution shown in Table 3.

The *attribute priority* and *preference value* are also randomly generated for each participant for each simulation run. For simplicity, our experiments do not cover commitment strategies that deal with concurrency effects since our computer simulation schedules the randomly generated meetings one at a time. Furthermore, it is assumed 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 Ref. [32]. In our simulations, since there is no concurrency, either *committed* or *non-committed strategies* can be used [32]. 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.

8.2. Measurements

To compare our algorithms, the following measurements were made on the simulation results:

H the number of hours to be scheduled

CD the number of occupied hours in the calendar per agent

S_{HC} the set of randomly generated meetings based on parameters H and CD

$|S_{HC}|$ the number of meetings in S_{HC}

$|S_{iHC}|$ the number of meetings in S_{HC} that involves agent i

M_{jHC} the jth meeting in S_{HC} to be scheduled (contains many alternative proposals)

PPt_{jHC} the number of participants in M_{jHC}

R_{jHC} the total number of requests required to schedule meeting M_{jHC}

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

$$R_{HC} = \left(\sum^{j} R_{jHC}\right)/|S_{HC}|$$

Measurements relating to committed meetings:

P_{ijHC} agent i's preference level for the final committed proposal for M_{jHC}

AP_{jHC} the average preference level (AP) for committed meeting M_{jHC} $AP_{jHC} = (\sum^{i} P_{ijHC})/PPt_{jHC}$

AP_{HC} the average of the AP's for all the committed meetings in set S_{HC}. Chart 3 shows a plot of this value for our experiments using the NWPI and NWOPI algorithms.

$$AP_{HC} = \left(\sum^{j} AP_{jHC}\right)/|S_{HC}|$$

Measurements relating to overall optimal solutions:

O_{jHC} the optimal or best AP for all proposals in meeting M_{jHC}

AO_{HC} the average of all the optimal AP's for the whole meeting set S_{HC}. Chart 1 shows a plot of this value for our experiments using the NWPI and NWOPI algorithms. $AO_{HC} = (\sum^{j} O_{jHC})/|S_{HC}|$

ADO_{jHC} the difference between the optimal AP and the AP of committed proposal for M_{jHC}

$$DO_{jHC} = O_{jHC} - AP_{jHC}$$

ADO_{HC} the average difference between the optimal AP and the committed AP for S_{HC}. Chart 2 shows a plot of this value for our experiments using the NWPI and NWOPI algorithms.

$$ADO_{HC} = \left(\sum^{j} DO_{jHC}\right)/|S_{HC}|$$

Measurements relating to an individual's expected solutions:

EP_{ijHC} agent i's expected preference for meeting M_{jHC}—the highest preference level out of all possible proposals for M_{jHC}

DE_{ijHC} agent i's difference between the expected and the final preference level for M_{jHC}

$DE_{ijHC} = EP_{ijHC} - P_{ijHC}$

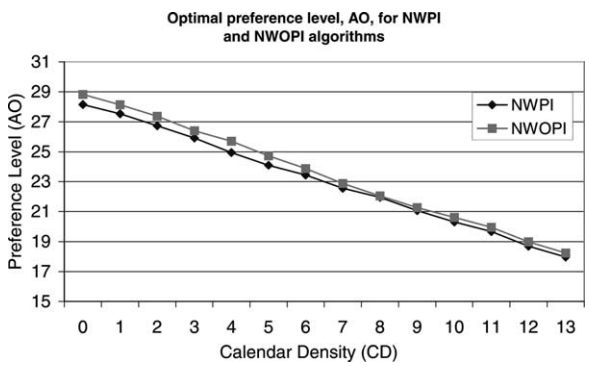

Chart 1. Optimal preference level for NWPI and NWOPI algorithms. This chart shows the computed average optimal values that will be used to determine whether optimality has been reached for the algorithms.

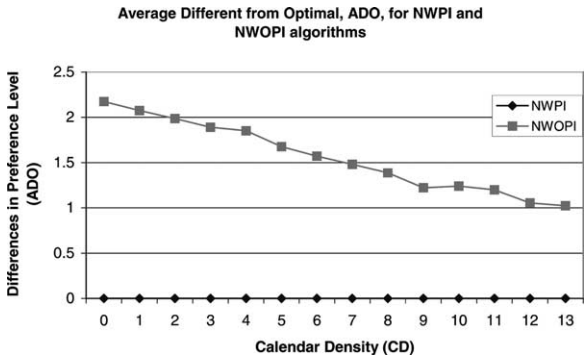

Chart 2. Average differences from optimal for NWPI and NWOPI algorithms. This chart verifies that NWPI truly finds optimal meeting schedules as the difference between computed optimal values is zero, whereas NWOPI differs slightly from optimal.

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

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

$\mathrm{ADE}_{\mathrm{HC}}$ the average difference from the expectation for the whole meeting set S_{HC}. Chart 4 shows a plot of this value for our experiments using the NWPI and NWOPI algorithms.

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

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

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

$\mathrm{APD}_{i\mathrm{HC}}$ the average percentage deviation of agent i for the meeting set S_{HC}. Charts 5 and 6 shows a plot of this value for our experiments using the NWOPI and NWPI algorithms respectively.

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

$\mathrm{APD}_{j\mathrm{HC}}$ the average percentage deviation for meeting $M\mathrm{j}HC$ for all participants

$$APD_{jHC}\left(\sum^{i} PD_{ijHC}\right)/PPt_{jHC}$$

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

$$APD_{HC}\left(\sum^{j} PD_{jHC}\right)/|S_{HC}|$$

To study the performance of our algorithms, two sets of computer simulations were carried out. Their results are presented in the following sections.

8.3. Simulation one

In this set of simulation experiments, we tested the performance of our algorithms purely on initial negotiation and without the CR mechanism. We obtained results over 100 simulation runs with the following input parameters: 35 'hours of meetings' and CD ranging from 0 to 13. The number of hours per meeting was designed such that each run has a similar number of meetings. We varied the CD parameter from 0 to 13, to compare the performance of the algorithms under different stress conditions (the higher the density, the fewer counter proposals and available intervals a participant will have). The CD plus H value must of course be less than or equal to the total calendar space.

We used two variations of the N* algorithm to test effects and benefits of using *preference estimation*: Algorithm 1 (NWOPI) simulates negotiation without preference indication and Algorithm 2a (NWPI) simulates negotiation with preference indication.

Algorithm 1. (NWOPI). NWOPI is similar to the negotiation process defined in [27], but instead of simply using the earliest timeslot to guide the negotiation, which might not be the most preferred, our user preference model is used. In this Algorithm, individual collaborating agent's preferences are not passed back to the initiating agent. Without preference information from other agents, the initiating agent will not be able to perform preference estimation and the whole negotiation process is guided by the personal preferences of the meeting initiator. This is similar to the *private preference model* of Ref. [11]. The initiating agent will terminate the negotiation when no mutually agreeable solution can be found as CR is not used in Simulation 1. The main purpose for this Algorithm is to simulate results produced by other researchers, such as Ref. [27], for comparison with our proposed N* algorithm.

The following action states show the main differences between Algorithm 1 (NWOPI) and the complete N* algorithm (the Step numbers refer to numbers in Fig. 7)

- *Step 4*. Select Proposal—the next n best proposals from the PV of the initiator will be selected, since the initiating agent only has the preference information of the initiator. Preference estimation is not used in the NWOPI algorithm. The original N* algorithm instead uses proposals from the PEV.
- *Steps 5*
  - *Step 5a*. Announce Proposal—just like N*, the selected proposals are announced to all the collaborating agents.
    - *Step C3*. Evaluate Proposal—each collaborating agent evaluates the proposals and returns a reply of either accept or reject for each proposal received. No preference or conflict levels are indicated,

unlike the full N* algorithm where the preference or conflict level will be passed back with the reply.
  - *Step C4*. Select Counter Proposals—just like N*, each collaborating agent tries to select *m* counter proposals that have higher preference levels than those proposed by the initiating agent.
- *Step 5b*. Get Replies—the proposal replies are updated in the *proposal table*. Unlike N*, no preference estimations will be performed as preference values are not passed back with the reply.
- *Step 5c*. Get Counter Proposals—the counter proposals are stored in the proposal table. Unlike N*, no preference estimations will be performed as preference values are not passed back with the reply.

- End Conditions:
  - *Case 1*: Impasse?—an impasse occurs when no more proposal to offer. In this case, there is no obvious solution and a meeting cancellation will be announced. Unlike N*, CR is not used in Algorithm 1 (NWOPI) and the original Step 6 is removed.
  - *Case 2*: Compromise?—terminates the negotiation if a feasible solution is found, otherwise continue the negotiation if proposals exist.

Algorithm 2a. (NWPI). This Algorithm extends Algorithm 1 with *preference estimation* which allows the initiating agent to be able to estimate the global preference level of each proposal. This Algorithm is basically identical to the proposed N* algorithm except that CR is not used. The following shows the main differences between this Algorithm (NWPI) and the original N* algorithm (shown in Fig. 7).

- End Conditions:
  - *Case 1*: Impasse?—an impasse occurs when any one agent rejects all proposals and has no counter proposal to offer. When this occurs, there is no obvious solution and the meeting will be cancelled. Unlike N*, the CR stage is not used in NWPI and Step 6 is removed for testing purposes.

8.4. Simulation two

In our second set of simulation experiments, we tested the effects and benefits of adding the CR algorithm. We obtained results over 100 simulation runs with the following input parameters: CD equals 10 and 'hours of meetings' to range from 29 to 59. In Simulation 2, the CD is fixed at 10 while the hours of meetings is varied instead. We also vary the *H* parameter from 29 to 59 to simulate congestion (when more and more meetings need to be scheduled and rescheduled). Three studies were performed: (1) the performance without CR under these same test conditions, (2) the performance of the CR algorithm and (3) the effect of using CE. Algorithm 2b (NWOCR) simulates negotiation without CR. Algorithm 3 (NWCR) simulates negotiation with CR but without CE. And finally, Algorithm 4 (NWEstCR) simulates the full N* algorithm. The performances of Algorithm 2b and 3 are first compared and then performances of Algorithm 3 and 4. In Simulation 2, the levels of participation are divided into must, should and can. An impasse occurs only when any of the participants that must or should participate is not available.

Algorithm 2b (NWOCR). This Algorithm is similar to Algorithm 2a and does not use CR. The main difference is that this Algorithm performs a more accurate *preference estimation* using the conflict level that is passed back as response by other collaborating agents.

Algorithm 3. (NWCR). This Algorithm is basically identical to Algorithm 2b, NWOCR, except this Algorithm uses CR when no solution can be found. The main difference between the Algorithm and the full N* algorithm is that there is no CE in this Algorithm.

The following shows the main differences between Algorithm 3 and the full version of the N* algorithm (Step numbers are those shown in Fig. 8).

- *Step 1* *. Find Least Conflict Set—the *least conflict set* is computed and sorted. The next proposal with the least conflicts is selected. Unlike the CR algorithm in N*, proposals need not check estimated conflicts, as this measure is not used in Algorithm 3 (NWCR).
- *Step 2* *. Announce Cancellation—this event will be cancelled if the least conflict set is empty (i.e. if all the proposals in the set have been tried and have failed).
  - *Step C1* *. Cancel—each collaborating agent will mark this event as cancelled.
- *Step 3* *.
  - *Step 3a* *. Announce Availability Check—just like N*, check if the collaborating agents are still willing to accept the proposals if the conflicts are resolved. Other events may have been scheduled since the point in time when the agents first reviewed the proposal, and other environmental changes may affect the proposal acceptance.
    - *Step C2* *. Check Availability—just like N*, each collaborating agent checks the proposal for acceptance.

  Step 3b *. Get Replies—the replies from the collaborating agents are consolidated. But unlike N*, CE is not performed.

Algorithm 4. (NWEstCR). This Algorithm is the full version of our proposed N* algorithm with preference

estimation, CR and CEs. This is basically identical to Algorithm 3, with the addition of CE.

8.5. Results from simulation 1

This section compares the results of Algorithm 1 (NWOPI) with those of Algorithm 2a (NWPI) obtained from Simulation 1. The main objective of Simulation 1 is to study the effects of N^* preference estimation in isolation from the CR algorithm. The simulation used the following parameters, which are similar to those used in [27].

Number of simulation cycles = 100
Hours of meetings, $H = 35$
Max number of participants = 6
Calendar size = 6
Length of day = 8
CD = from 0 to 13
Number of proposals = 1
Number of counter proposals = 1
All participants must participate the meeting
All meeting with the same priority

Charts 1 and 2 summarize key measurements for Algorithm 1 (NWOPI) and Algorithm 2a (NWPI) with the average results obtained over 100 simulation runs with the following input parameters: hours of meetings = 35, and CD ranging from 0 to 13. The average optimal preference level, AO_{HC}, and the average difference from optimal, ADO_{HC}, of the two algorithms in different calendar densities are shown in Charts 1 and 2, respectively. The AO_{HC} is the average of the highest possible preference level that can be found for each meeting schedule attempt. This value is slightly lower for Algorithm 2a, as it always finds an optimal solution; hence, the optimal preference value for each additional 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 may not have been taken. As the table verifies, Algorithm 2a (NWPI) does indeed always finds an optimal solution.

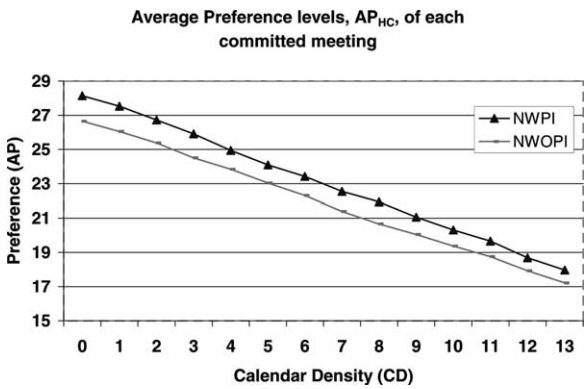

Chart 3. The average preference levels of each committed meeting. NWPI always finds better solutions, i.e. those with higher preference levels.

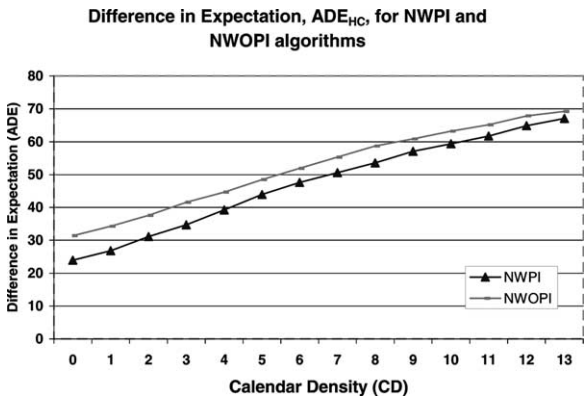

Chart 4. The difference in expectation for NWPI and NWOPI algorithms. Solutions produced by NWPI are closer to the highest individual preference level than NWOPI. Therefore these solutions will be more likely to be accepted by the users.

Chart 3 shows that the average preference levels of the committed meeting schedules obtained by Algorithm 2a (NWPI) are always higher than those found by Algorithm 1 (NWOPI) over all calendar densities. In other words, Algorithm 2a (NWPI) always finds better solutions.

Furthermore, Chart 4 shows the difference in expectation ADE_{HC} for Algorithms 1 and 2a. In simple terms, this measure shows the difference between the preference levels of the committed meeting schedules compared with the highest possible individual preference levels. We found that the Algorithm 2a (NWPI) minimizes this difference. This means users should be happier with results generated by Algorithm 2a (NWPI) compared with those generated by Algorithm 1 (NWOPI).

Next, tests were performed to examine whether the 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 5 and 6 show the average percentage deviation of preference levels from what was expected for each participant. Chart 5 shows that Algorithm 1 (NWOPI) has a strong bias for Agent 0

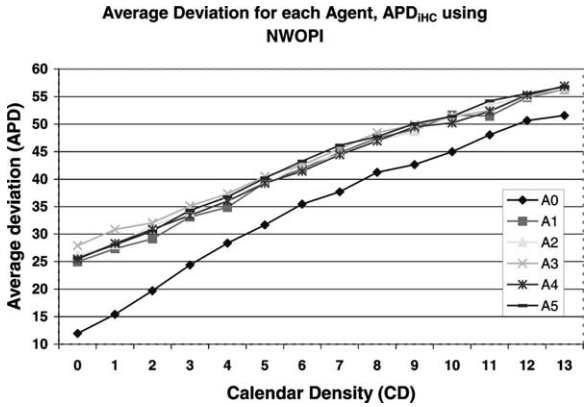

Chart 5. The average deviation for each agent using Algorithm 1 (NWOPI). This shows Algorithm 1 works in favor of A0, the meeting initiator.

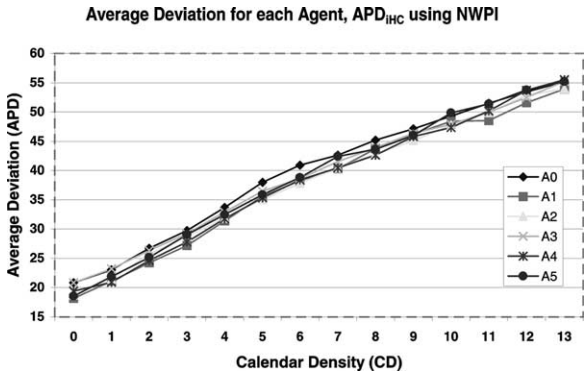

Chart 6. The average deviation for each agent using Algorithm 2a (NWPI). This shows Algorithm 2a is pretty much neutral to all participants of the meetings.

(A0)—the meeting initiator. The chart shows that A0's deviations are much smaller than others. On the other hand, Chart 6 shows that Algorithm 2a (NWPI) produces similar deviations for each agent; each agent is treated equally in terms of matching their preference expectations. Hence, in Algorithm 1 (NWOPI) the initiating agent has an advantage over other participants. 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.

We also computed the average rates for successfully scheduling meetings as well as the number of cycles required to schedule a meeting for different CD (see Table 4). We found that on average, Algorithm 2a (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 2a finds an optimal solution but requires more negotiation cycles. It is interesting to note that the success rate for Algorithm 2a is actually slightly higher than Algorithm 1. In other words, finding optimal solutions actually increases the likelihood of success.

Table 4
The success rate and communication cost

CD	Success rate		Cycle	
	Alg 1	Alg 2a	Alg 1	Alg 2a
0	0.947923	0.954405	2.524374	10.13024
1	0.924794	0.93429	2.811912	10.09234
2	0.899848	0.90594	3.198848	10.3727
3	0.873886	0.877417	3.549373	10.72629
4	0.846403	0.841522	3.857161	11.11498
5	0.812604	0.813839	4.262623	11.63381
6	0.785587	0.785447	4.604638	12.13455
7	0.754743	0.75642	4.880225	12.52017
8	0.726248	0.733714	5.191055	12.73768
9	0.700973	0.706197	5.216777	13.19788
10	0.681579	0.678945	5.425814	13.23528
11	0.657313	0.658303	5.566396	13.51758
12	0.62766	0.624302	5.770078	14.01227
13	0.604402	0.601903	5.691775	14.07976

Results from Simulation 1 show that by using preference levels sent back by collaborating agents, the average deviation for each agent could be minimized without bias towards any particular agent. Moreover, use of the preference estimation technique provides an optimal solution but requires more negotiation cycles.

8.6. Results from simulation 2

The main objective of this simulation exercise was to study the performance of CR and the effect of CE. The following simulation parameters were used:

Number of simulation cycles = 100
Hours of meetings, H = form 29 to 59
Priority of meetings: from 1 to 7
Max number of participants = 6
Participation level: must, should, can
Calendar size = 6
Length of day = 8
CD = 10
Number of proposals = 1
Number of counter proposals = 1

To compare Algorithm 2b (NWOCR) and Algorithm 3 (NWCR), we used the success rates of the two algorithms and the distribution of success with respect to meetings of different priorities. Chart 7 shows the success rates of the two algorithms.

Chart 7 shows that the success rate of Algorithm 3 (NWCR) is higher than that of Algorithm 2b (NWOCR), which did not utilize CR. From Chart 8, it can be determined that with CR the success rate is significantly increased, by 18.9% on average.

Furthermore, the effect of CR on meetings with different priorities was tested. Charts 9 and 10 show the success rate for meetings with different priorities. Chart 9 shows that Algorithm 2b (NWOCR) produced similar success rates for meetings with different priorities, while Chart 10 shows that in Algorithm 3 (NWCR), higher priority meetings (highest priority is P7 in the charts) have higher success rates. This is probably because lower priority events might be cancelled during CR in order to release resources for higher priority events. This is probably similar to the handling of high priority events in real life.

Next, the performance of Algorithm 3 (NWCR) is compared with the proposed N^* algorithm, i.e. Algorithm 4 (NWEstCR), which includes the preference estimation, CR and CE techniques. In this experiment, we tested the effects of CE. Algorithm 3 (NWCR) is similar to Algorithm 4 but without CE. The main objective in using CE is to reduce the number of interactions and the communication costs during negotiation in order to identify a feasible solution or be able to terminate the negotiation earlier (reduce communication)

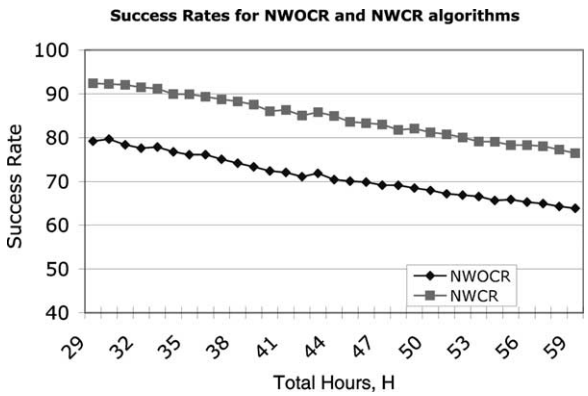

Chart 7. The success rates for Algorithm 2b (NWOCR) and Algorithm 3 (NWCR). This demonstrates that Algorithm 3 was able to successfully schedule more meetings than Algorithm 2b.

if the algorithm can predict when no solution can be found. Thus, we compare the number of interactions needed in Algorithm 4 (NWEstCR) with that of Algorithm 3 (NWCR). Chart 11 shows the average number of requests requiring CR and Chart 12 shows the percentage improvement of the proposed algorithm in terms of communication cost.

From Chart 11, it was determined that, compared to NWCR, use of CE in the proposed N* algorithm can reduce the number of requests in completing a negotiation while maintaining the same success rate. In other words, the proposed N* algorithm performs just as well as Algorithm 3 but faster.

Furthermore, Chart 12 shows that Algorithm 4 actually performs even better when there are more meeting hours, H, to schedule! As the total number of meeting hour increases, the number of meetings to be scheduled will also increase. When the number of meetings increases, the number of failures will increase and the interaction saved in CR by Algorithm 4 becomes more significant.

We further compare the communication cost of the above four algorithms in terms of number of negotiation cycles. Chart 13 shows the average number of cycles required to complete a negotiation on different CD with the average results obtained over 100 simulation runs with

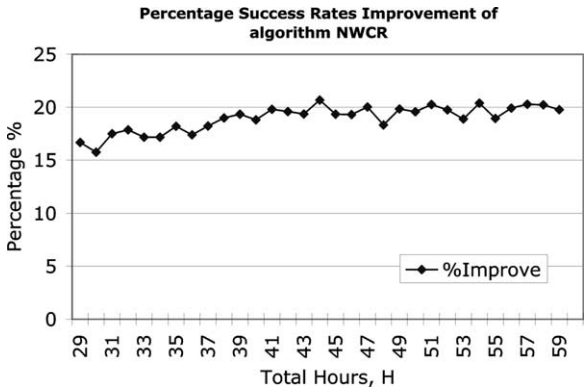

Chart 8. The percentage success rate improvement of Algorithm 3 (NWCR). Algorithm 3, on average, performs roughly 18.9% better than Algorithm 2b in successfully scheduling meetings.

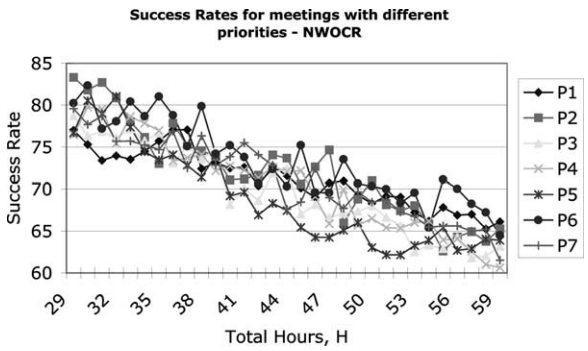

Chart 9. The success rates for meetings with different priorities—NWOCR. Using NWOCR, meetings with different priorities have similar success rates.

the following input parameters: hours of meetings = 35, and CD ranging from 0 to 15. The results show that Algorithm 2b (NWOCR) requires less cycle to complete compare to Algorithm 2a (NWPI) because Algorithm 2b has a more accuracy estimation on participant preference and is able to predict earlier when there is no solution. It is interesting to note that with better preference estimation (using both preference and conflict levels) the number of cycle required for Algorithms 1 and 2b are very close for cases with high calendar densities. Moreover, we can find that the number of cycles required by Algorithm 3 (NWCR) increases when the CD increases. It is because more cycles are required to perform CR as the higher the density, the more conflicts there will be. Comparing Algorithm 4 (NWEstCR) to Algorithm 3, the number of cycles is further reduced by CE. Furthermore, by comparing results of Algorithm 4 with Algorithm 2a, we found that Algorithm 4 requires less negotiation cycles than Algorithm 2a. This is because (1) Algorithm 4 uses a more accuracy preference estimation (the same reason why Algorithm 2b is better then Algorithm 2a), (2) performs CE, which can predict earlier when there is no solution (in Algorithm 2a, the initiating agent will try all possible proposals), and (3) requires only those participants that must or should attend be available.

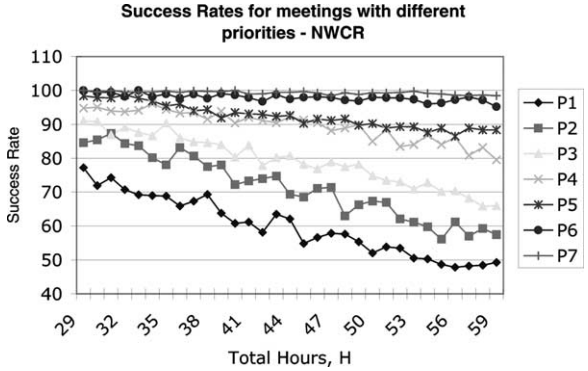

Chart 10. The success rate for meetings with different priorities—NWCR. Using NWCR, meetings with higher priorities have better success rates.

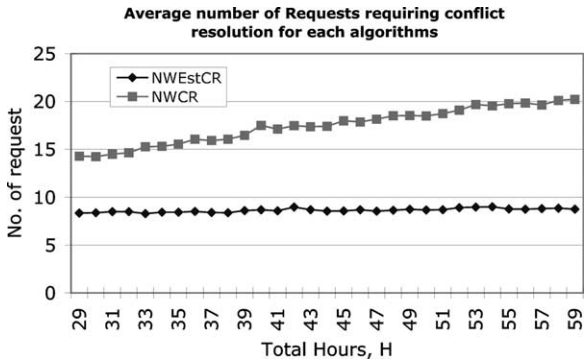

Chart 11. No of requests requiring CR for NWCR and NWEstCR. The NWEstCR algorithm requires less communication between agents.

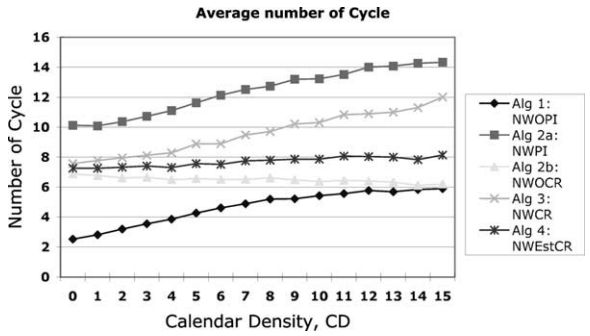

Chart 13. The number of negotiation cycle in different CD. Our proposed N* algorithm (NWEstCR) produces higher quality solutions in less time than all the other variations we have tested.

9. Applications

The N* negotiation algorithm is a general negotiation algorithm that can be used for potentially different types of applications. In this paper, we used an example of agent-based meeting scheduling, which is part of our MAFOA (mobile agents for office automation) environment [38,39]. MAFOA is a research project to design a comprehensive set of intelligent agents that can support different aspects of office automation, such as purchasing, workflow, meeting scheduling, etc. The main objective is to free people from mundane and routine tasks so that they can concentrate on actual productive work. The MAFOA environment was implemented using Java and the IBM Aglet toolkit [18,19].

The N* negotiation algorithm can also be used for resource allocation problems. For example, we are exploring how N* can be used for airport stand allocation [6]. In this problem, each aircraft arriving at an airport will need to be assigned a stand or gate, in addition to other resources such as baggage handlers, ramp personnel, airline arrival staff, cleaning staff, catering, etc. We represent each aircraft arrival as a *negotiation event*. The participants or collaborating agents are the various service providers and airport resources, such as the physical stand/gate. Besides the airport authority and airline, other agencies or companies might be providing some of the supporting services. Each of these service providers may have their own unique set of preferences and constraints. N* will allow a schedule to be produce that maximizes the business and operational objectives of all the participants.

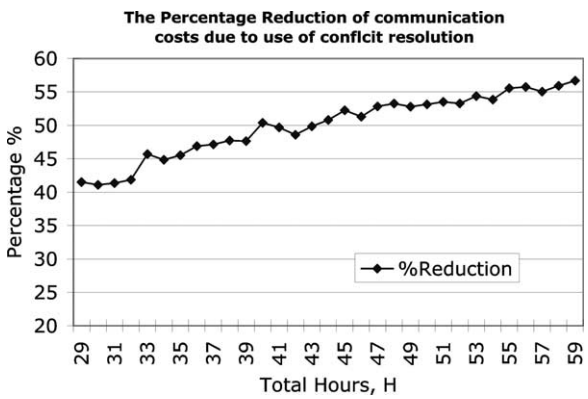

Chart 12. Percentage reduction of communication costs due to use of CE. On average, NWEstCR reduces communication costs by one half.

Bouzid and Mouaddib [3] used agent-based negotiation for another type of transportation scheduling problem—the scheduling of trucks to orders. This is a distributed problem since trucks might be owned and operated by different companies, each with their own set of business and operational rules and constraints. Orders come in and are scheduled dynamically, as in our N* algorithm. [3] proposes an approached that made use of *fuzzy characteristics functions* and an *extended contract net protocol*. Fuzzy temporal characteristics are used to deal with uncertainty in agent behaviors. Another application of distributed agent-based negotiation for transportation can be found in Ref. [10], where there is *vertical negotiation* between trucks and *horizontal negotiation* between companies.

10. Conclusion

In this paper, we presented our distributed negotiation framework and the N* negotiation algorithm. The framework consists of a user preference model and an approach that makes use of the model to evaluate proposals and suggest counter proposals, using a preference level measure. The negotiation process is guided by a proposal evaluation function that evaluates the global preference level for a particular proposal. The N* negotiation algorithm is based on A* and finds the optimal solution that maximizes average preference levels. N* resolves impasse, problems with no mutually agreeable solution, using a CR process that is similar to dependency directed backtracking. We have tested our N* algorithm on the classical meeting scheduling problem using a computer simulation test bed that simulates the scheduling of hundreds of randomly generated meetings.

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

[1] Albrecht K, Albrecht S. Added value negotiating: the breakthrough method for building balanced deals. IL, USA: Irwin Professional Publishing; 1993.
[2] Biswas J, Bhonsle S, Wee TC, Yong TS, Weiguo W. Distributed scheduling of meetings: a case study in prototyping distributed applications. Proceedings of the Second International Conference on Systems Integration, ICSI'92; 1992. pp. 656–665.
[3] Bouzid M, Mouaddib A-I. Cooperative uncertain temporal reasoning for distributed transportation scheduling. Proceedings of the International Conference on Multi Agent Systems; 1998. pp. 397–398.
[4] Bradshaw JM, editor. Software agents. Cambridge, MA: MIT Press; 1997.
[5] Bui HH, Venkatesh S, Kieronska D. A multi-agent incremental negotiation scheme for meetings scheduling. Proceedings of the Third Australian and New Zealand Conference on Intelligent Information Systems, ANZIIS-95; 1995. pp. 175–180.
[6] Chun HW, Chan S, Tsang F, Yeung D. Stand allocation system (SAS)—a constraint-based system developed with software components. AI Magazine 2000;21(4):63–74.
[7] Conry SE, Meyer RA, Lesser VR. Multistage negotiation in distributed planning. In: Bond AH, Gasser L, editors. Readings in distributed artificial intelligence. San Mateo: Morgan Kaufman; 1998. p. 367–84.
[8] Ferber J. Multi-agent systems: an introduction to distributed artificial intelligence. Reading, MA: Addison-Wesley; 1999.
[9] Finin T, Labrou Y, Mayfield J. KQML as an agent communication language. In: Brad-shaw JM, editor. Software agents. Cambridge, MA: AAAI Press/The MIT Press; 1997. p. 291–316. Chapter 14.
[10] Fischer K, Chaib-draa B, Muller JP, Pischel M, Gerber C. A simulation approach based on negotiation and cooperation between agents: a case study. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 1991;29(4):531–45.
[11] Garrido L, Sycara K. Multi-agent meeting scheduling: preliminary experimental results. Proceedings of the Second International Conference in Multi-Agent Systems (ICMAS'96), Kyoto, Japan; 1996.
[12] Guttman RH, Maes P. Agent-mediated integrative negotiation for retail electronic commerce. Proceedings of Workshop on Agent Mediated Electronic Trading, AMET-98; 1998.
[13] Guttman RH, Maes P. Cooperative vs. competitive multi-agent negotiations in retail electronic commerce. Proceedings of the Second International Workshop on Cooperative Information Agents (CIA'98), Paris, France; 1998.
[14] Higa K, Shin B, Sivakumar V. Meeting scheduling: an experimental investigation. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics 1996;3:2023–8.
[15] Huhns MN, Stephens LM. Multiagent systems and societies of agents. In: Weiss G, editor. Multiagent systems: a modern approach to distributed artificial intelligence. Cambridge, MA: MIT Press; 1999.
[16] Jennings NR, Jackson AJ. Agent-based meeting scheduling: a design and implementation. Electronics Letters 1995;31(2):350–2.
[17] Jeong WS, Yun JS, Jo GS. Cooperation in multi-agent system for meeting scheduling. Proceedings of the IEEE Region 10 Conference (TENCON 99), vol. 2.; 1999. p. 832–835.
[18] Lange D, Oshima M. Programming and deploying Java mobile agents with aglets. Reading, MA: Addison-Wesley; 1998.
[19] IBM Japan—Aglets Workbench: http://www.trl.ibm.co.jp/aglets/
[20] Lawler EL, Wood DW. Branch and bound methods: a survey. Oper Res 1966;14:699–719. ORSA.
[21] Liu JS, Sycara KP. Distributed meeting scheduling. Proceedings of the Sixteen Annual Conference of the Cognitive Science Society, Atlanta, Georgia; 1994.
[22] Luo X, Leung H-F, Lee JH-M. A multi-agent framework for meeting scheduling using fuzzy constraints. Proceedings of the Fourth International Conference on MultiAgent Systems; 2000. p. 409–10.
[23] Park S, Birmingham WP. Multiagent Negotiation Framework. Technical Report (CSE-TR-248-95), University of Michigan, Ann Arbor; 1995.
[24] Pino JA, Mora HA. Scheduling meetings with guests' approval. Proceedings of the XVII International Conference of the Chilean Computer Science Society; 1997. p. 182–189.
[25] Pino JA, Mora HA. Scheduling meetings using participants' preferences. Info Technol People 1998;11(2):140–51.
[26] Schwartz R, Kraus S. Negotiation on data allocation in multi-agent environments. Artif Intell 1997;94(1–2):79–98.
[27] Sen S, Durfee EH. A formal study of distributed meeting scheduling group decision and negotiation. Group Decision Negotiat Support Syst 1998;7:265–89.
[28] Sen S, Haynes T, Arora N. Satisfying user preferences while negotiating meetings. Int J Hum–Comput Stud 1997;47:407–27.
[29] Sen S. Developing an automated distributed meeting scheduler. IEEE Expert 1997;12(4):41–5.
[30] Sen S, Durfee EH. A contracting model for flexible distributed scheduling. Ann Oper Res 1996;65:195–222.
[31] Sen S, Durfee EH. On the design of an adaptive meeting scheduler. Proceedings of the Tenth IEEE Conference on Artificial Intelligence for Application, San Antonio, Texas; 1994. p.40–6.
[32] Sen S, Durfee EH. The role of commitment in cooperative negotiation. Int J Intell Cooper Inform Syst 1994;3(1):67–81.
[33] Smith RG. The contract net protocol: high-level communication and control in a distributed problem solver. IEEE Trans Comput 1980;C-29(12):1104–13.
[34] Stallman RM, Sussman GJ. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. Artif Intell 1977;9(2):135–96.
[35] Shapiro SC, editor. Encyclopedia of artificial intelligence. New York: Wiley; 1992.
[36] Tsuchiya K, Takefuji Y, Kurotani K, Wakahara K. A neural network parallel algorithm for meeting schedule problems. Proceedings of the 1996 IEEE TENCON'96, Digital Signal Processing Applications, vol. 1.; 1996. p. 173–177.
[37] Weiss G, editor. Multiagent systems: a modern approach to distributed artificial intelligence. Cambridge, MA: MIT Press; 2000.
[38] Wong RYM, Ho ATT, Fung SKL, Chun AHW. A model for resource negotiation using mobile agents. Proceedings of 4th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2000), Orlando, Florida; 2000.
[39] Wong RYM, Chun HW. Optimizing user preferences while scheduling meetings. Proceedings of the Third International Conference on Enterprise Information Systems, (ICEIS 2001), Setubal, Portugal; 2001.