Appl Intell (2006) 24: 143–154
DOI 10.1007/s10489-006-6935-y

Meetings scheduling solver enhancement with local consistency reinforcement

Ahlem Ben Hassine · Tu Bao Ho · Takayuki Ito

Abstract Meeting scheduling (MS) represents an important real-world group decision application that denotes one of the actual combinatorial problems. Solving this problem consists of scheduling all the meetings while satisfying all the constraints related to both the users and the meetings. However, given human nature, the solution is usually delineated by the encountering of conflicting preferences. Most of existing research efforts allow the relaxation of the users' preferences in order to reach an agreement between all the participants, which is not always possible. In addition, they do not deal with the achievement of any level of local consistency to enhance the efficiency of the solving process, and finally, they do not address the real difficulty of distributed systems, which is the complexity of message passing operations.

Here we propose a new approach to facilitate and streamline the scheduling meetings process in any organization. This approach is based on the *distributed reinforcement of arc consistency* model, which takes into account the difficulties mentioned above. The present work focuses mainly on satisfying meetings hosts' preferences as much as possible, while taking into consideration all users' availability. The underlying *selfish* protocol is able to efficiently reach the best solution for the host of the meeting (according to the predefined criteria) whenever possible. This process is achieved with the minimal number of exchanged messages and while retaining as much of the privacy of the involved users as possible. An experimental comparative analysis divulges that our approach is scalable and worthwhile especially for strong constraints.

Keywords Meeting scheduling problems · Valued distributed constraint satisfaction problems · Local consistency enforcement · Multi-agent system

1. Introduction

In daily life, meeting scheduling (MS) is a pre-eminent and typical group decision support problem. It can be described by the process of scheduling events (meetings) involving individual constraints, i.e., private preferences over alternative mutual decisions. These constraints are crucially related to the availabilities and preferences of the users who should participate in the meetings. The MS problem is naturally distributed and cannot be solved by a centralized approach. Solving it involves determining the date, the time, place and the duration of meetings[1] that must be held among several users, depending on their available times, calendar, and preferences. Moreover, each meeting has a priority that indicates its degree of importance. Therefore, solving an MS problem means to find an optimal solution (satisfying some predefined optimality criteria) whenever possible. Various optimality criteria have been suggested in game theory, economics and voting theory. Thus, the solver process should seek for a compromise among different human users' requirements

A. B. Hassine (✉) · T. B. Ho
School of Knowledge Science, Japan Advanced Institute of Science and Technology,
1-1 Asahidai, Nomi-shi, Ishikawa-ken, Japan
e-mail: {hassine, bao}@jaist.ac.jp

T. Ito
Graduate School of Engineering, Nagoya Institute of Technology,
Gosiko, Showa-ku Nagoya, 466-8555, Japan
e-mail: itota@ics.nitech.ac.jp

[1] To simplify the problem, in the rest of this paper, we use the term date to define the date, time and duration of a meeting, while for the place we assume that all participants are in the same city.

regarding both the potential meetings' times and meetings' priorities.

Many significant research efforts dealing with the MS problem have been proposed in the literature; among them, [1, 5, 7, 11, 12, 13, 20, 22] will be outlined in the next section.

The basic common points stressed in the majority of works in the literature are:

- The relaxation of any of the users' constraints, even non-availability constraints, to achieve agreement between all of the agents and consequently to solve the problem. However, in the real world problems, it is not always possible to relax the constraints of users. For example, if a user has to travel for business, then such a constraint would oblige him/her to cancel the trip in order to attend the meeting, which may not be always possible.
- None of these works has considered the high complexity of message passing operations in real distributed systems.
- Only the authors in [10, 11] have dealt with the use of some inferred knowledge to maintain the coherence between meetings in order to steer the selection of the next proposal; none of the other efforts tried to maintain any level of consistency [14] during the negotiation process despite the pre-eminent role of filtering techniques in solving NP-Complete problems. These techniques allow the simplification of any constraint problems by eliminating values or a combination of values that cannot be involved in any solution. Integrating the enforcement of local consistency within the search process is worthwhile for pruning inconsistent values, and consequently saving much fruitless exploration of the search tree, especially on hard and large problems leading to an enhancement of the solving process.

We have learned from all the previous work and focused our research essentially on a new multi-agent MS approach that closely reflects real applications while improving the process of scheduling meetings. The proposed protocol is based on a *distributed reinforcement for arc consistency* (DRAC) approach [2]. This approach is an asynchronous agent-based approach to enforce arc consistency on any binary constraint network. Note that several levels of local consistency [14] exist in the literature (node, arc, path, k-consistency). Nevertheless, arc consistency (AC) is the most used one due to its low cost.[2]

The basic idea of this work is to benefit from the main goal of DRAC in order to reduce the complexity of the meeting-scheduling problem solving process. Thus, the meeting-scheduling problem is contemplated as a set of distributed reactive self-interested agents in communication; each of them acts on behalf of one user, and maintains its personal information in private. All the agents cooperate and negotiate by exchanging only relevant information. Their decisions are based only on current available local knowledge and without necessarily having any global view. Thus the final result, i.e., the scheduling of the meetings maintained by agents, is obtained as a consequence of the agents' interactions. In such a manner, all of the agents act in parallel and asynchronously via the sending of asynchronous point-to-point messages. This paper significantly extends our previous work [3].

In this work, we propose to formalize the MS problem as a *valued constraint satisfaction problem* (VCSP) [17] in which each user maintains two kinds of constraints: hard and soft constraints related to themselves, in addition to the other strong constraints defining the problem. The hard constraints (which can never be violated) represent the non-availability of the user, while the soft constraints (which can be violated) represent the preference calendar of a user. Furthermore, each new scheduled event is considered as a hard constraint.

This paper is organized as follows. In Section 2, we outline some of the related works. In Section 3, we give the proposed formalization for the MS problem. In Section 4, we present the DRAC model adapted to the MS problem. In Section 5, we describe the global MS dynamic. In Section 6, we discuss the termination and complexity properties. In Section 7, we give the experimental results. Finally, Section 8 concludes the paper.

2. Related works

Among the research studies dealing with solving MS problems are those based on CSP (constraint satisfaction problem) formalism [15]. The underlying problem is formalized as centralized CSP in which all the users' information is centralized in the same process [1, 5]. These works are essentially focused on over-constraint CSPs.

However, recent researchers have argued that the best way to solve MS problems is to use an agent-based approach. The main reason is that agents can accomplish their tasks through cooperation while allowing the users to keep their privacies. A mechanism design approach based on multi-agent system (MAS) to solve MS problems was reported first by Ephrati et al. [7]. The authors defined two paradigms of MS scenarios, o*pen scheduling systems* and *closed scheduling systems*. In our work we focus on the second type,[3] in which the participants belong to a same body, e.g., an organization or

[2] Higher levels of local consistency are more costly and some of them may change the structure of the constraint network.

[3] Indeed, the proposed approach can be applied also to an open scheduling system, where all participants do agree to attend any meeting if the time is possible.

a company. The authors proposed three scheduling mechanisms for the closed system, which differ in the information type that each user has to reveal about his individual preferences. They tried to approximate the optimal utilitarian choice while avoiding manipulability by using the Clark Tax method [8]. Garrido and Sycara [12] reported another MAS work that focused on using distributed autonomous and independent agents to solve the problem. Each agent has its individual goal, to schedule the meeting while maximizing its individual preferences. This work is based on the communication protocol presented in [18] where in agents are capable of negotiating and relaxing their constraints in order to reach an agreement on a schedule with high joint utility.

Sen et al. [20] have proposed another technique based on how an application domain for intelligent surrogate agents can be analyzed, understood and represented in order to make these agents able to carry out tasks on behalf of human users, taking into account their environment. Their prior work has focused on agents adapting to environmental changes [21], but in [20] their efforts were directed towards the integration of user preferences. Often users' preferences are mutually conflicting, so the authors used techniques from voting theory to formally represent and reason with conflicting preferences.

Three other multi-agent approaches to MS problems, using the Partial CSP formalism introduced by [9], appear in the literature. The first work proposed by [13] offered a new approach to MS problems using fuzzy constraints. The underlying protocol is called the selfish protocol, in which each user tries to maximize his/her preferences during the negotiation process. The second, in [22], used the *distributed valued constraint satisfaction problem* (DVCSP) formalism to model the MS problem. This approach is used in our experimental evaluation.

The third work, based on multi-agent systems and using fuzzy constraints to express users' preferences, was presented by Franzin et al. [11]. This meeting scheduling system was based on an existing system that includes hard constraints [10]. The authors proposed, in their work, to integrate preferences to their system and focused on observing the behavior of this new system under several conditions [11]. Their main objective was to evaluate the relations among solution quality, efficiency and privacy. The correlated protocol is based on inferring some new knowledge during the solving process, which may clash with the desires of agents to keep their information private. Two basic global optimization criteria, considered in this work were fuzzy optimality and Pareto optimality. However, in this protocol the number of exchanged messages increases with the size of the problem (number of possible proposals and users). In addition, in the worst case each agent has to reveal all its proposals in order to reach optimality.

3. Proposed MS problem formalization

We propose to formalize the MS problem as a VCSP (valued constraint satisfaction problem) [14]. Therefore, we first present the CSP formalism followed by VCSP formalism and then we give our proposed formalization for the MS problem.

A CSP [15] is a triplet (X, D, C) composed of a finite set of n variables $X = \{X_1, \ldots, X_n\}$, each of which takes values in an associated finite domain $D = \{D_1, \ldots, D_n\}$ and a set of e constraints among these variables $C = \{C_{ij}, \ldots\}$; C_{ij} is a binary constraint between X_i and X_j. The constraints restrict the values the variable can simultaneously take. Solving a CSP consists of finding one or all-complete assignments of values to variables satisfying all the constraints.

This formalism is generalized to the over-constrained problems by giving a weight or a valuation to each constraint reflecting the importance of satisfying it. A VCSP [17] is a quintuple (X, D, C, S, φ) where (X, D, C) is a classical CSP formalism, $S = (\mathrm{E}, \otimes, \phi)$ is a valuation structure and φ: $C \rightarrow$ E. E is the set of possible valuations; ϕ is a total order on E; $\perp \in$ E corresponds to the maximal satisfaction, and $\otimes$ is an aggregation operator used to aggregate valuation.

Assume that A is an assignment of all the variables of the problem. The valuation of A is defined by:
$\varphi(A) = \otimes_{c \in C} \varphi(A, c)$ where:

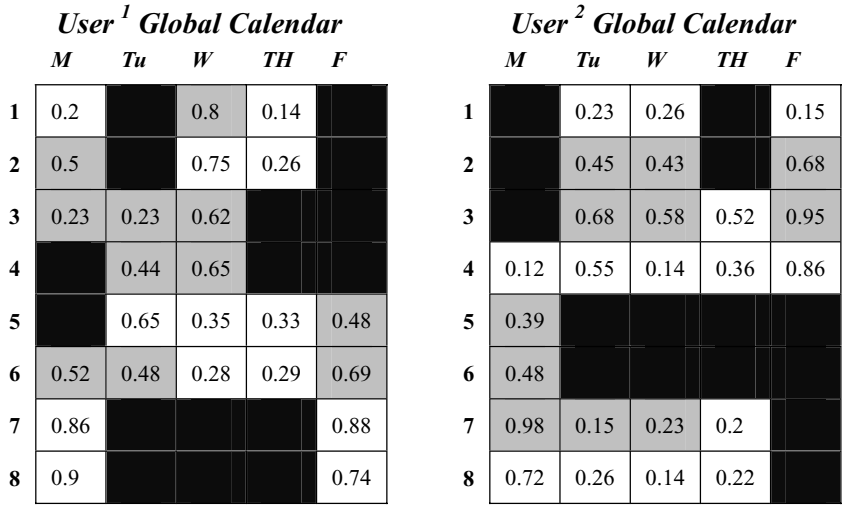
$$\varphi(A, c) = \begin{cases} \perp \text{ If } c \text{ is satisfied by } A \\ \varphi(c) \text{ Otherwise} \end{cases}$$

We have used the VCSP formalization to define the MS problem by (X, D, C, S, φ), in which:

- $X = \{\ldots, X_k^i, \ldots, X_h^j, \ldots\}$, where X_k^i is the date requested by user $user^i$ for the meeting m_k^i.
- $D = \{\ldots, D_k^i, \ldots, D_h^j, \ldots\}$, where D_k^i is the set of possible dates for the meeting m_k^i.
- C is the set of all the constraints of the problem. We divide set C into two types of constraints: constraints related to the users and constraints related to the meetings. For the former, we can consider:
  - Hard constraints: C_H related to the non-availability of the users,
  - Soft constraints: C_S related to the preferences of the users.

With regard to the second type of constraints, it represents all the *allDiff* constraints [19] existing between each pair of meetings sharing at least one of the same participants.

Fig. 1 "Examples of the calenders of two users"



It is noteworthy that for this type all the constraints are considered as hard constraints.

– S is the valuation structure where $\otimes$ corresponds to $+$, and ϕ defines the operator more than ">". This operator is used to set a total order among the obtained solutions in the problem.

For each hard constraint $c_H \in C_H$, we associate a weight $\perp$, for each soft constraint $c_S \in C_S$ we associate a weight[4] $W^i_{kl} \in E$, and to each meeting m^i_k we associated a weight[5] $w^i_k \in E$ ($E = [0..1]$).

To illustrate this formalization more clearly, let us consider the following example formed by 2 users, each entrusted with the task of scheduling one meeting. Assume that both meetings require the participation of all the users. Figure 1 illustrates the preferences of each user. The underlying MS formalization (X, D, C, S, φ) is as follows:

– $X = \{X^1_1, X^2_1\}$,
– $D = \{D^1_1, D^2_1\}$, D^i_k is represented by the gray boxes in Fig. 1, i.e., possible times for the underlying meeting.
– $C = C_H \cup C_S$ where:

  - C_H is represented by the black boxes in Fig. 1. and all the *allDiff* constraints existing between each pair of meetings ($X^1_1 \neq X^2{}_1$).
  - C_S is represented by the white boxes in Fig. 1.

The number inside the boxes indicates the degree of preferences of each user for each time in their calendar.

[4] It represents the degree of preference of the agent A_i for having the meeting m^i_k at the date d_{kl}.

[5] This weight defines the priority/importance of m^i_k.

4. DRAC model adapted to the MS problem

The DRAC model uses two kinds of agents: Constraint agents and an Interface agent. Each has its own knowledge (static and dynamic), a local behavior to satisfy, and a *mailbox* to store incoming messages. The agents communicate by exchanging asynchronous point-to-point messages. An agent can send a message to another only if it knows the other belongs to its acquaintances. For transmission between agents, we assume that the messages are received in a finite delivery time and in the same order they are sent. Messages sent from several agents to a single one may be received in any order.

The Interface agent is an intermediate interface between all the Constraint agents. It is added in order to create the agents and, most importantly, to inform the users of the result.

This model can be "well" adapted to the MS problem. In this problem, each Constraint agent can be considered as a User agent A_i. A User agent must maintain the concerned user's calendars for his/her availability, preferences and the already planned meetings.

The acquaintances of an agent consist of all of the agents that must be present in the same meeting, called Participant agents (represented as $Part(m^i_k)$). Accordingly, in our system an agent is considered as a Proposer agent when it has a meeting to schedule. It can be also considered as a Participant agent if it is a participant in another meeting proposed by another agent of the system. Each scheduled meeting that has been registered (represented as $Calendar^{Ai}$) is considered as a new constraint. Therefore it must be added to the set of hard constraints maintained by the corresponding agents.

Each agent A_i maintains a VCSP^{Ai} for which the variables $X^{Ai} \in X$ represent the meetings dates to found for its user's set of meetings (represented as $Meetings^{Ai}$), while the constraints $C^{Ai} \in C(C^{Ai} = C^{Ai}_H \cup C^{Ai}_S \cup C^{Ai}{}_{allDiff})$ represent the non-availability, the preferences of the corresponding

user, the timetabling of this user and the constraints relating to each pair of its meetings.

Thus in the proposed model, the aforementioned constraints represent the intra-agent constraints for A_i while the inter-agent constraints are represented by a set of strong constraints, i.e., equality constraints. An equality constraint exists between agents A_i and A_j if and only if at least one meeting m_k^i (resp. m_h^j) exists, such that $A_j \in Part(m_k^i)$ (resp. $A_i \in Part(m_h^j)$). It is noteworthy that the inter-agent constraints are dynamic because the participants and their number in a meeting differ from one meeting to another.

Each attendant has a set of meeting preferences for each particular meeting. The local goal is to schedule meetings such that all its hard constraints C_H are satisfied while trying to maximize the proposer's preferences (*selfish* protocol) using formula 1.

$$Max \sum_{k\in\{1\ldots|meetings(A_i)|\}} W_{k_l}^i \tag{1}$$

The global goal is to schedule the maximum of users meetings satisfying all the inter-agent constraints.

In this paper, we consider the MS problem from standpoint of the host of each meeting (who can be the director of the company, the manager of the department, etc.). In our scenario, we will adopt the natural, innately fair and self centered behavior of a human being, since the knowledge of a user is self centered knowledge. Hence, each agent in the meeting scheduling process tries to satisfy its local goal while maximizing its preferences (*selfish* protocol).

The adopted criteria for an MS solver should guarantee some common attributes for both the resulting decision and the scheduling process itself. Furthermore, in order to ensure such features for the solver and the outcomes, the proposed system should be able to extract the truthful preferences [7] and availabilities of the users.

In the present work, the optimal solution is based on self-centered initiator preferences. Overbidding the preferences for any time cannot change the outcome; therefore the dominant strategy for every agent is to reveal its utility values truthfully. Obviously, the optimality criteria may differ from one scenario to another according to the measures adopted by the system designer, e.g., a pure utilitarian approach [7], the Nash approach [16], and others, while the global proposed dynamic remains the same for any chosen measurement. In the case of a global measurement, e.g., maximizing the summation of the participants' utilities, we propose as stated in [7] to use convenience points to express preferences over alternative times for every proposed new meeting. In addition, we propose to embed the Clark Tax mechanism [8] to force the users to truly express their preferences towards the meeting's importance and users' possible timings. As for the non-availability of the participant, we assume that the users will reveal their real availability if necessary.

Hence, each participant in a meeting will get a total number of convenience points to spread among its availability-times according to its preferences and this for each new meeting added to the system. The Tax can be computed on the number of convenience points given for the next meeting.

5. Global MS Dynamic

The global objective of the proposed approach is to schedule all meetings while maximizing the hosts' local preferences. In addition, we focused on minimizing the total number of exchanged messages. The multi-agent meeting scheduling negotiation protocol is divided into two steps as follows:

- The first step uses the basic idea of the DRAC approach, which consists in transforming the original MS problem into another *equivalent* MS'. This step is needed to reinforce some level of local consistency [14] (node and arc consistency) in the initial problem.
- The second step solves the obtained MS problem while maintaining arc-consistency and this is accomplished via interactions and negotiations between Participant agents and the Proposer agent. Each Proposer agent searches for the best solution for its meetings that, on the one hand, fulfils the condition given in the previous section, and on the other, satisfies all hard constraints.

When a user wants to host a meeting, he has to run the Interface agent, which will activate the corresponding Proposer agent and make it interact with all of the Participant agents (Fig. 2). More than one Proposer agent can be activated at the same time, in the case of multiple users who want to schedule their meetings.

Each activated Proposer agent must first reduce the time slots of the corresponding meetings according to its hard

```
Begin
1.  For each m^i_k ∈ Meetings^Ai do
2.    For each d_k1∈D_k^Ai such that
        d_k1∈C_H^Ai OR ∃ m^j_h / m^j_h∈Calendar^Ai
        and X_h^Aj = d_k1 do
3.          D_k^Ai ← D_k^Ai ¥ d_k1;
4.  If D_k^Ai=∅
5.    Then change calendar D_k^Ai of m^i_k;
6.    Else For each A_j∈ Part(m^i_k) do
7.         Send(A_j,self,
           "ReduceCalendar:D_k^Ai for:m^i_k");
End
```

Fig. 2 "*Start* process executed by each Proposer agent A_i".

Fig. 3 "Main procedures executed by each agent A_i"

```
ReduceCalendar:D for:m
1. For each d∈D such that
     d∈C_H^Ai OR ∃ m^j_h / m^j_h∈Calendar^Ai
       and X_h^Aj = d do
2.    D ← D ¥d;
3. Send(Sender, self, "Reply:D for:m");
End

Reply:D for:m^i_k
4. SetD ← setD ∪ D;
5. If Size(SetD)=|Part(m^i_k)|

6.   Then  D ← D ⋂_{i∈{1..|SetD|}} SetD[i];

7. If D=∅
8.   Then Change m^i_k possible times;
9.   Else Choose d∈D/satisfy formula(1);
10.    For each A_j∈ Part(m^i_k) do
11.      Send(A_j,self,"ReceiveProposal:d for:m^i_k");
End

ReceiveProposal:d for:m
12. res ← true;
13. If (∃ m^j_h∈Calendar^Ai and X_h^Aj = d)
14.   Then res ← false;
15. If (res=true)
16. Then Add(Calendar^Ai, (m^j_h, d));
17.      Update set C_H;
18. Send(Sender,self,"Response:res for:m");
End

Response:res for:m
19. setRep ←setRep ∪ res;
20. If Size(SetRep)=|Part(m^i_k)|
21.   Then If ∃SetRep[i],i∈{1.. |SetRep|}/
              SetRep[i]=false;
22.          Then Choose another date d';
23.            For each A_j∈ Part(m^i_k) do
24.              Send(A, self,
                 "ReceiveProposal:d' for:m^i_k");
25.          Else For each A_j∈ Part(m^i_k) do
26.                Send(A, self,"Confirmation:d for:m^i_k");
End
```

constraints, constraints defining the non-availability of the user. This process can be viewed as a local reinforcement of node consistency and aims to reduce the meetings' slot times by eliminating the dates upon which the meeting cannot be held (Fig. 2, lines 1 and 2). In other words, a meeting cannot be held on a date defined as a non-available date for the user or already planned for another meeting.

If the time slots for a meeting become empty after reduction that indicates that the corresponding user is not available for all of the proposed dates of this meeting, the times of this meeting must then be changed (Fig. 2 line 5). Otherwise, the Proposer agent must send the obtained reduced time slots for all of the meetings to be scheduled to all of the Participant agents (Fig. 2 line 7).

Each Participant agent that receives this message starts first by eliminating both the non-viable dates from the received time slots of the meetings (dates that correspond to its non-availability), and all the dates taken by already scheduled meetings (Fig. 3 lines 1 and 2). After that, it returns the obtained time slots to the sender agent (Fig. 3 line 3). At first, the Proposer agent collects all the received reduced slot times (Fig. 3 line 4), then, begins by scheduling its meetings. It tries to first find the proposal that maximizes its preferences (Fig. 3 lines 6 and 9) and then sends it to the concerned acquaintances. If the Proposer agent cannot find a solution to this problem, then it changes the time slots of this meeting (Fig. 3 line 8).

Each agent that receives this proposal must first check if it has, meanwhile, accepted another proposal for the same

date. In the negative case, the agent will first update its hard constraints by adding the new proposal (Fig. 3 line 16), then update the dates of its not-yet-scheduled-meetings by eliminating the dates that correspond to the same date of the just scheduled meeting, in order to maintain the arc-consistency. Finally it informs the Proposer agent of its agreement (Fig. 3 line 18).

However, if the agent has another meeting already scheduled at the same time as the proposed meeting, it must send a negative answer to the Proposer agent and ask it to change its proposal (Fig. 3 line 14). Accordingly, each agent that has proposed a meeting and received at least one negative answer must change its proposal (Fig. 3 line 21). Consequently, this agent must decrease its degree of preferences (Fig. 3 line 22) and the same process is repeated until an agreement is reached among all of the participants. If after testing all of the solutions no agreement is reached, then the Proposer agent is obliged to inform the participants of the meeting cancellation.

The aforementioned dynamic resumes running until the system reaches its stable equilibrium state. This state can be defined as the satisfaction of all agents in the system. The satisfaction of an agent is defined as the scheduling of all its meetings or the cancellation of the ones that cannot be held at that time.

We should emphasize the fact that in this paper we assume on the one hand that each newly scheduled meeting will be considered as a hard constraint, and on the other hand, each agent performs a selfish protocol. This choice is used in order to avoid dynamic changes and especially to escape from an infinite processing loop. This work can be considered as the first version of the proposed approach. The integration of the dynamic process was presented in [4] where the proposed protocol focused on maximizing the utility of all the agents of the system.

6. Discussions

6.1. Termination

The dynamic of the MSRAC approach ends when the system reaches its stable equilibrium state. In real application, this state will be temporary, and the whole system will restart with new sets of meetings to schedule. However, at the stable equilibrium state all the agents are satisfied; that satisfaction is defined for each agent by two aspects, the completion of scheduling all its current meetings, and the acquisition of all the confirmations from all the other Proposer agents. However, this approach is guaranteed to find a useful solution, i.e., the best one for the Proposer, if it exists. The host of each meeting will check all possible dates from the most preferred to the less preferred one to schedule its meeting. Nevertheless, to prove the termination of this approach we have to prove that the underlying protocol never goes into an infinite loop while scheduling a meeting.

Let's assume that this approach goes into an infinite loop while scheduling a meeting. To schedule a meeting X_l^i all the participants will cooperate together to find the best date for this meeting. The system will go into an infinite loop while scheduling X_l^i if and only if the Participant agents reprocess the checked dates (cycle) when no solution is found. However, the number of possible meeting dates is discrete and finite. Moreover, every unsuccessfully checked date is removed from the system to avoid returning to it later. The system will stop when a "good" date is found or when all possible dates for X_l^i are processed and no possible solution has been found. Hence our assumption is not true.

We have to note that the satisfaction state of all the agents in a distributed system can be achieved by taking snapshots of the system, using the well-known algorithm of Chandy-Lamport [6]. Termination occurs when all agents are waiting for a message and there is no message in the transmission channels. The cost of the termination process can be mitigated by combining snapshot messages with our protocol messages.

6.2. Complexity

Let us consider an MS problem implying n for total number of users, d for the maximal number of possible dates per meeting, $|C_H| = c_H$ for the maximum number of preferred dates per user and $|C_S| = c_S$ for the maximum number of non-available dates per user. The total number of agents in this system is n the same as the total number of users. Suppose that each meeting involves n attendees and each user has m already scheduled meetings in his/her calendar. Let's compute the complexity of scheduling an another meeting into the existing schedule of each agent.

The solving process of the proposed scenario is divided into two steps. In the first step, the pruning step, the initiator agent will perform $O((c_H + m)d)$ operations to filter the time slots of the new event. Then, this agent will transmit the obtained set of possible remaining dates to the $(n-1)$ attendees to carry out the same process. The time complexity of this step, in the worst case, is $O(nd(c_H + m))$.

For the second step, the initiator agent will first determine the intersection of the sets of the received times, leading to $(n-1)d^2$ operations, then choose one proposal among d dates,[6] according to the proposed optimal criteria; this process requires $O(c_S d \log(d))$ operations. The agent will send its proposal to the attendees to check it. Each

[6] In the worst case, we assume that no possible date is removed from the initial set.

attendee will perform O(m) operations to check if it has received in the meanwhile another proposal for the same date. The same process resumes until exhausting all possible dates, leading to O(dnm) operations in the worst case. Then the total temporal complexity of the second step is $O((n-1)d^2 + c_S d\log(d) + dnm)$.

Finally, the temporal complexity for each event to schedule is, $O(nd(c_H + m) + nd^2 + c_S dlog(d) + dnm)$ in the worst case.

The spatial complexity for all the agents is $O(n(c_S + c_H + d + m))$ in the worst case, where $(c_S + c_H + d + m)$ is the total size of the initial calendar for each agent.

7. Experimental comparative evaluations

To evaluate the proposed approach, we have developed the multi-agent dynamic with Actalk, an object oriented concurrent programming language using the Smalltalk-80 environment. In our experiment, we generated random meeting scheduling problems varying from simple to hard problems. The parameters used for a meeting problem are: n agents in the system, m meetings per agent, p participants in a meeting, D global calendar, c_H number of hard constraints per agent,c_S number of initial soft constraints per agent, d maximal possible dates per event, W_{kl}^i weights for the soft constraints, and w_k^i weights of the meetings (the weight of each hard constraint is equal to 1).

In order to compare our approach with that reported in [22], we used the same parameters to run both algorithms on randomly generated examples. We must note that the approach in [22] presents some restrictions on, first, the handle of the hard constraints (i.e., all the constraints could be relaxed by this approach), and second, the discrimination between meetings. This approach processes all the proposed meetings with the same importance independently of neither the Proposer nor the attendants. However in the real world, all meetings are not equivalent. For this reason we have included the notion of meeting priority in our formalization by associating a weight w_k^i to reflect its greatness. Our approach tries then, in its solving process, to *first* schedule the most important meeting maintained by each agent, unlike the approach in [22]. In that manner, we attempt to describe ideally the real world meeting scheduling problems. Therefore two kinds of experimentation are given in this section.

For the first kind, we assume that for each generated problem, we have only soft constraints. We carried out the two approaches on the same meeting instances with: $n = 10, m \in \{3, 4, 5\}$, $p = 7$, $c_S \in \{20, 40, 60\}$; $W_{kl}^i \in [0..1]$, $w_k^i \in [0..1]$ were randomly chosen (10 instances are generated for each $< m; c_S >$). The initial calendar in each problem is equal to 60.

Table 1 "Ratio of mean results of the CPU time for meeting problem without hard constraints (10 instances were generated for each $<m; c_S>$)".

	<3; 20>	<4; 20>	<5; 20>	<3; 40>	<4; 40>
Ratio CPU	0.75	0.73	1.01	1.60	2.46
	<5; 40>	<3; 60>	<4; 60>	<5; 60>	
Ratio CPU	4.15	3.99	5.88	7.77	

Table 1 shows the obtained mean results for the ratio of CPU time of the approach in [22] divided by the CPU time of ours. In order to analyze these results, let us consider the two cases <3; 20> and <4; 20>. At first glance, it seems that the approach in [22] is better than our approach, given that it requires less CPU time in these two cases. However, this can be justified by the fact that for this kind of problem the constraints are few, so the approach in [22] can rapidly find a solution for each meeting without relaxing many constraints, causing few iterations on the same meeting.

In our approach, on the other hand, we try to find the solution that maximizes the user's preferences (not the "first" solution). Therefore we must check all the possible dates for each meeting. In addition, in our approach we try to maximize each user's preference for the most important meetings. In all the other cases, the approach in [22] takes more time than our approach as both the number of constraints and the number of meetings grows. Furthermore, in our approach each agent tries to perform all its meetings in parallel while for the approach in [22], it is done in a sequential manner.

As the number of meeting constraints grows, so does the probability of getting the same dates for the meetings. Therefore, the number of relaxed constraints by the approach in [22], increases. This leads to additional iterations for the same meeting and hence an increase in the CPU time.

As for the second kind of experimentation, i.e., to appraise the greatness of the reinforcement of local consistency in the solving process, we have chosen to measure the percentage of reduction made by the first step of our approach. For this purpose, examples including hard constraints were randomly generated with $n = 10, m = 3$, $p \in \{7, 5, 3\}$, $c_H \in \{20, 30, 40, 50\}$ and $d \in \{66\%, 50\%, 33\%, 16\%\}$ corresponding respectively to each c_H. For each pair $< p, c_H >$, 10 instances were generated; we then ran each instance 10 times and measured the average of the achieved results. These results are expressed in terms of four criteria: (i) the CPU time spent by each of the two approaches, (ii) the percentage of scheduled meetings, (iii) the percentage of reduced soft constraints performed by the first step of the

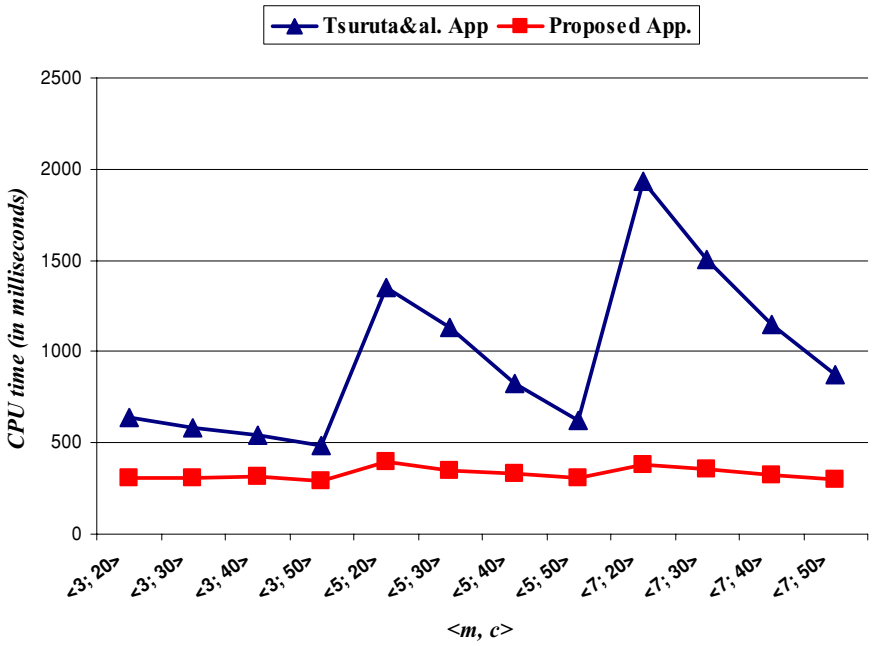


Fig. 4 "Mean results in term of CPU time (10 instances were generated for each $< p; c_H >$)"

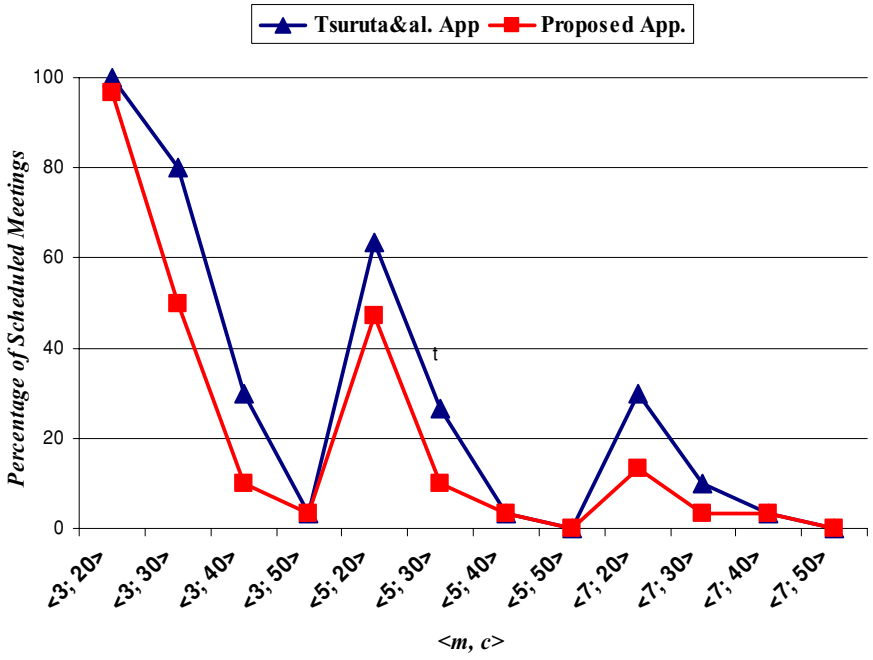


Fig. 5 "Mean results in terms of the percentage of scheduled meetings (10 instances were generated for each $< p; c_H >$)"

proposed approach and (*iv*) the required number of messages passed.

We have introduced some modifications to the approach in [22] to make it worthwhile for both hard and soft constraints. These two approaches were carried out on the same meeting examples. Figures 4 and 5 show the achieved mean results of both approaches in term of CPU time and the percentage of scheduled meetings.

These results show that our approach requires less CPU time than approach [22]. For example in the case of 7 participants and 50 hard constraints, the problem is over-constrained and thus no meetings can be planned, i.e., no agreement can be reached between all the attendants. Therefore, our approach can discover merely the absence of solution from the first step, and before starting the solving process.

In case of 7 participants and 20 hard constraints, only a few meetings can be planned. Table 2 shows that the percentage of pruned dates from possible ones is high (= 96, 42%). Therefore, our approach is able to schedule the possible

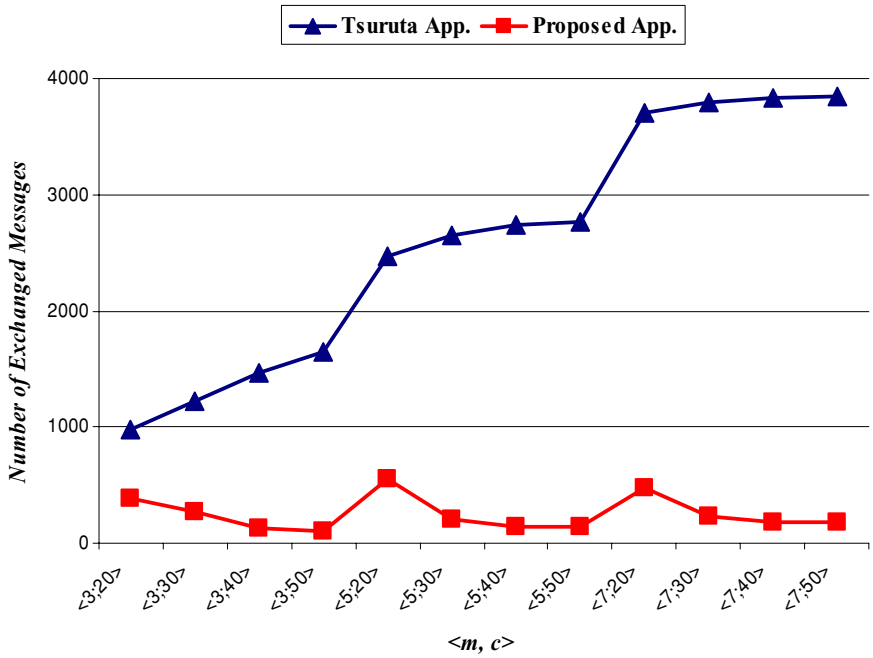


Fig. 6 "Mean results in term of the number of exchanged messages (10 instances were generated for each $< p; c_H >$)"

Table 2 "Mean results in term of the percentage of reduced time slots".

	<3; 20>	<3; 30>	<3; 40>	<3; 50>
% C_H *Reduction*	80.83	94.5	98.5	99.67
	<5; 20>	<5; 30>	<5; 40>	<5; 50>
% C_H *Reduction*	91.75	98.67	99.83	100
	<7; 20>	<7; 30>	<7; 40>	<7; 50>
% C_H *Reduction*	96.42	99.22	99.83	100

meetings in considerably less CPU time than approach [22], i.e., the approach in [22] requires about four times the time needed by our approach. This result can be elucidated by the fact that the first step is useful in order to discard the dates that cannot be in any solution and consequently avoid exploiting them in the solving process, leading to decreased CPU time consumption.

Nevertheless, for the percentage of the meetings scheduled, the approach in [22] planned, for some cases, more meetings than our approach. This is because for our approach we tried to plan the most important meeting first. For example, in the case <7; 20> the 30% of the meetings scheduled by the approach in [22] may or may not contain the most important meetings in the problem. Meanwhile, with our approach we are sure that the 13.33% of the meetings scheduled are the most important because they were first chosen to be processed using their weights.

As for the number of exchanged messages needed to reach an agreement among all the users, Fig. 6 shows that the proposed approach requires many fewer exchanged messages than the Tsuruta approach [22]. This number increases with the number of participants in the meeting, even if the problem has no solution, i.e., there is no possible time at which all the participants can meet. However, with the proposed approach, the percentage of reduced values increases with both the number of participants and the number of hard constraints, consequently the number of exchanged messages decreases.

We can conclude that our approach is a scalable approach that outperforms the approach proposed in [22] and this is especially true when the number of meetings increases. One must note also that our approach seems to be more appropriate to real-world applications by dealing especially with strong constraints (i.e., inequality) and by bringing forward consideration of discrimination among the proposed meetings. In addition, the first step of the proposed approach can fulfill a premature detection of the impossibility of reaching any agreement between all the participants; it does this by maintaining arc-consistency.

8. Conclusion

The objective of this paper is to propose a new approach for meeting scheduling (MS) problems that reflects real-world applications. To fulfill such condition, we have considered, in our model, two kinds of constraints to model the users' requirements: hard constraints to model the non-availability of a user and soft constraints to define his/her preferences.

The underlying multi-agent architecture associates an agent with each user and makes the agents interact by sending asynchronous point-to-point messages containing only relevant information to keep, as much as possible, their privacy. The basic idea of this approach consists of two steps: the

first reduces the initial problem by reinforcing some level of local consistency (node and arc consistency); and the second step solves the resulting meeting scheduling problem with a minimum number of exchanged messages.

This approach was implemented with Actalk under the Smalltalk-80 environment and compared with an existing approach in literature [22] on randomly generated instances, in mean of CPU time, percentage of scheduled meetings and the number of exchanged messages. The obtained results show that our approach is scalable and worthwhile for processing strong constraints. In addition, they show the importance of the first, i.e., the reduction step. Other experiments were made to measure the percentage of non-viable values discarded from the meetings' calendars. The obtained results showed that this process is appropriate for reducing the MS problem, and consequently the search space, without loss of solutions.

In future work, we will try to improve this approach by allowing a more relaxed system. Then, we will try to implement the improved approach on a multi-processor platform using real data.

Acknowledgment

This work is supported, in part, by the Japanese Foundation for C&C Promotion Grant for non-Japanese researchers to Ahlem Ben Hassine.

References

1. Abdennadher S, Schlenker H (1999) Nurse scheduling using constraint logic programming. In: Proc. IAAI, pp 838–843
2. BenHassine A, Ghédira K (2002) How to establish arc-consistency by reactive agents. In: Proc. ECAI, pp 156–160
3. BenHassine A, Ito T, Ho TB (2004) Scheduling meetings with distributed local consistency reinforcement. In Lecture Notes in Artificial Intelligence of the IEA/AIE, pp 708–717
4. BenHassine A, Défago X, Ho TB (2004) Agent-based approach to dynamic meeting scheduling problems. In: Proc. 3rd Inter. Joint Conference AAMAS, pp 1132–1139
5. Brakker RR, Dikker F, Tempelman F, Wognum PM (1993) Diagnosing and solving over-determined constraint satisfaction problems. In: Proc. IJCAI, pp 276–281
6. Chandy KM, Lamport L (1985) Distributed snapshots: determining global states of distributed systems. TOCS 3(1): 63–75
7. Ephrati E, Zlotkin G, Rosenschein JS (1994) A non-manipulable meeting scheduling system. In: Proc. Inter. Workshop on Distributed Artificial Intelligence, pp 105–125
8. Ephrati E, Rosenschein JS (1991) The clarke tax as a consensus mechanism among automated agents. In: Proc. the 9th National Conference on Artificial Intelligence, pp 173–178
9. Freuder EC, Wallace RJ (1990) Partial constraint satisfaction. Artificial Intelligence 58(1–3):21–70
10. Franzin MS, Freuder EC, Rossi F, Wallace R (2002) Multi-agent meeting scheduling with preferences: efficiency, privacy loss, and solution quality. In: Proc. AAAI Workshop on Preference in AI and CP
11. Franzin MS, Freuder EC, Rossi F, Wallace R (2004) Multi-agent meeting scheduling with preferences: efficiency, solution quality and privacy loss. Computational Intelligence 20(2):264–286
12. Garrido L, Sycara K (1996) Multi-agent meeting scheduling: preliminary experimental results. In: Proc. ICMAS, pp 95–102
13. Luo X, Leung HF, Lee JH-M (2000) Theory and properties of selfish protocol for multi-agent meeting scheduling using fuzzy constraints. In: Proc. ECAI, pp 373–377
14. Mackworth AK (1970) Consistency in network of relations. Artificial Intelligence 8:99–118
15. Montanari U (1974) Networks of constraints: fundamental properties and applications to picture processing. Information Sciences 7:95–132
16. Nash JF (1953) Two-person cooperative games. Econometrica 21:128–140
17. Schiex T, Fargier H, Verfaillie G (1995) Valued constraint satisfaction problems: hard and easy problems. In: Proc. 14th IJCAI, pp 631–637
18. Sycara K, Liu JS (1994) Distributed meeting scheduling. In: Proc. 16th Annual Conference of Cognitive Society
19. Stergiou K, Walsh T (1999) The difference all-difference makes. In: Proc. IJCAI, pp 414–419
20. Sen S, Haynes T, Arora N (1997) Satisfying user preferences while negotiating meetings. Inter. Journal of Human-Computer Studies 47:407–427
21. Sen S, Durfee EH (1994) On the design of an adaptive meeting scheduler. In: Proc. 10th IEEE Conference on AI Applications, pp 40–46
22. Tsuruta T, Shintani T (2000) Scheduling meetings using distributed valued constraint satisfaction algorithm. In: Proc. of ECAI, pp 383–387

Ahlem Ben Hassine received a B.S and M.S. degrees from the High Institute of Management of Tunis (ISG), in 1996 and 1999. She received her PhD degree in 2005 from the School of Knowledge Science at the Japan Advanced Institute of Science and Technology (JAIST). Her main research interests include artificial intelligence mainly for solving complex problems, constraint satisfaction problems, multi-agent systems, meeting scheduling problems.

Takayuki Ito received a B.Eng., M.S., and PhD degrees from Nagoya Institute of Technology (NIT) in 1995, 1997 and 2000. He is an Associate Professor at the Graduate School of Engineering, NIT. His research interests include computational mechanism design, auction, agent-mediated electronic commerce, multi-agent negotiation, agent-based group decision support systems, truth maintenance systems, and knowledge representation.

Tu Bao Ho received a B. Eng. degree from Hanoi University of Technology in 1978, M.S. and PhD. Degrees from University Paris 6, in 1984 and 1987, a Habilitation diploma in 1998 from University Paris Dauphine. He is currently a professor at School of Knowledge Science, Japan Advanced Institute of Science and Technology (JAIST). His research interests include artificial intelligence, machine learning, knowledge based systems, knowledge discovery and data mining.