Interacting with the User in Calendar Management

Liliana Ardissono, Giovanna Petrone, Marino Segnan, and Gianluca Torta

Dipartimento di Informatica, Università di Torino, Italy
{liliana.ardissono, giovanna.petrone, marino.segnan, gianluca.torta}@unito.it

Abstract. The management of personal and shared schedules is limited by the lack of flexibility of time management services. Online calendar managers fail to support reasoning about time and multi-user activities. Task schedulers fully control the temporal allocation of events without involving their users in decision making. As an attempt to address such issues, this paper presents a mixed-initiative scheduling model supporting interactive calendar management. Our model is applied in MARA (Mixed-initiAtive calendaR mAnager), which enables users to solve scheduling problems by incrementally exploring the solution space.

Keywords: mixed-initiative scheduling, temporal reasoning.

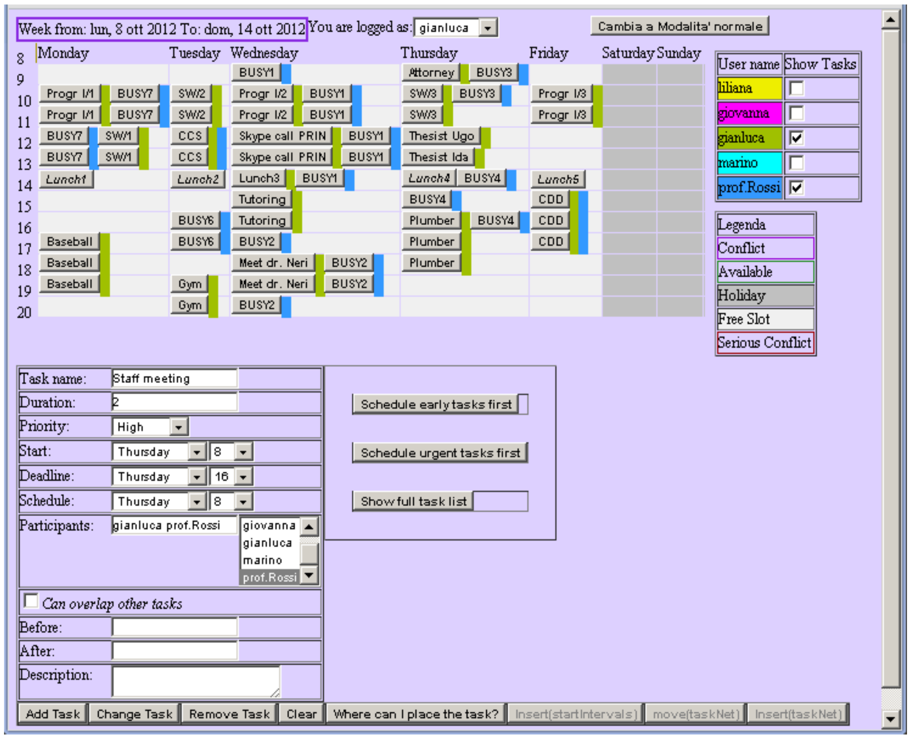
1 Introduction

Current on-line calendar managers allow calendar sharing. However, they provide limited support in the organization of non trivial schedules because they lack the capability to reason about temporal constraints. Indeed, calendar management becomes challenging when schedules are overconstrained or include items having multiple participants because a possibly large number of constraints has to be analyzed to find any feasible solutions. Therefore, an automatic support is desirable for this type of work.

In the past, fully or semiautomated scheduling systems were proposed to handle calendars. However, such systems "fail to account for the personal nature of scheduling, or they demand too much control of an important aspect of an individual's working world" [1]. Calendar managers should interact with the user enabling her/him to schedule activities in a personal way in order to play the role of user-friendly support tools.

We propose to address this issue by exploiting Intelligent User Interfaces to mediate the dialog between user and scheduler, enabling the user steer the calendar management. Thanks to the adoption of a mixed-initiative interaction model, the user can incrementally explore the available scheduling options while being aware, at each stage, of their pros and cons. Thus, (s)he can follow the preferred path, guiding the scheduling process and benefiting from the help of the system in possibly complex time reasoning tasks.

The work described in this paper proposes a new, mixed-initiative scheduling model that enables the user to temporally allocate multi-user events and tasks *in cooperation* with the system. The paper also proposes a novel, *conservative scheduling policy* to suggest calendar revisions by modifying small portions of the schedules, leaving the rest as originally planned or with minor temporal shifts. The idea is that of keeping the changes to the user's plans as local as possible in order to maintain stable daily plans.


Cambia a Modalita' normale
Schedule early tasks first
Schedule urgent tasks first
Show full task list
Add Task
Change Task
Remove Task
Clear
Where can I place the task?
Insert(startIntervals)
move(taskNet)
Insert(taskNet)

Week from: lun, 8 ott 2012 To: dom, 14 ott 2012 You are logged as: gianluca

8	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
9			BUSY1	Attorney BUSY3			
10	Progr I/1 BUSY7	SW/2	Progr I/2 BUSY1	SW/3 BUSY3	Progr I/3		
11	Progr I/1 BUSY7	SW/2	Progr I/2 BUSY1	SW/3	Progr I/3		
12	BUSY7 SW/1	CCS	Skype call PRIN BUSY1	Thesist Ugo			
13	BUSY7 SW/1	CCS	Skype call PRIN BUSY1	Thesist Ida			
14	*Lunch1*	*Lunch2*	Lunch3 BUSY1	*Lunch4* BUSY4	*Lunch5*		
15			Tutoring	BUSY4	CDD		
16		BUSY6	Tutoring	Plumber BUSY4	CDD		
17	Baseball	BUSY6	BUSY2	Plumber	CDD		
18	Baseball		Meet dr. Neri BUSY2	Plumber			
19	Baseball	Gym	Meet dr. Neri BUSY2				
20		Gym	BUSY2				

User name	Show Tasks
liliana	☐
giovanna	☐
gianluca	☑
marino	☐
prof.Rossi	☑

Legenda
Conflict
Available
Holiday
Free Slot
Serious Conflict

Task name:	Staff meeting
Duration:	2
Priority:	High
Start:	Thursday 8
Deadline:	Thursday 16
Schedule:	Thursday 8
Participants:	gianluca prof.Rossi (giovanna, gianluca, marino, prof.Rossi)
☐ *Can overlap other tasks*	
Before:	
After:	
Description:	

Fig. 1. User interface of our mixed-initiative scheduler (working days view).

Our scheduling model is applied in the MARA Mixed-initiAtive calendaR mAnager, which supports the management of online shared calendars including multi-user activities. MARA exploits Temporal Constraint Satisfaction Problem techniques for reasoning about time. A preliminary test with users provided encouraging results on the efficacy of MARA's features. Section 2 overviews MARA and describes a use case. Section 3 presents the temporal reasoning underlying the system. Sections 4 and 5 describe the evaluation results and the related work. Section 6 concludes the paper.

2 Overview of MARA

MARA supports cross-calendar scheduling by reasoning on the temporal constraints associated to calendar items. Figure 1 shows a user's calendar in a week of October 2012. The table in the upper right portion of the page shows the list of shared calendars and enables the user to select those to be jointly visualized. In order to let the user identify the participants of the events, each item has associated a number of vertical bars whose colors correspond to those of the respective calendars; e.g., the participants of event CDD, scheduled on Friday afternoon, are Gianluca and Prof. Rossi.

The lower portion of the page supports the creation/deletion/revision of calendar items (see the form at the left side and the Add/Change/Remove Task buttons below it).

Moreover it enables the user to revise the calendar by selecting the scheduling policy to be applied (“Schedule early tasks first”, “Schedule urgent tasks first”); see [2].

2.1 Example

Gianluca, a University staff member, collaborates with some colleagues (Liliana, Giovanna, Marino) and with the Head of Department, Prof. Rossi. Gianluca’s calendar includes teaching hours (e.g., Progr I, SW), Department meetings (CDD, CCS), student support (Tutoring, Thesist Ugo / Ida), project meetings (e.g., Skype call PRIN, Meet Dr. Neri) and personal commitments (e.g., Baseball, Plumber, Attorney). Some activities have strict schedules; e.g., teaching hours and Department meetings. Others are flexible; e.g., Gianluca could meet students on Wednesday, Thursday and Friday.

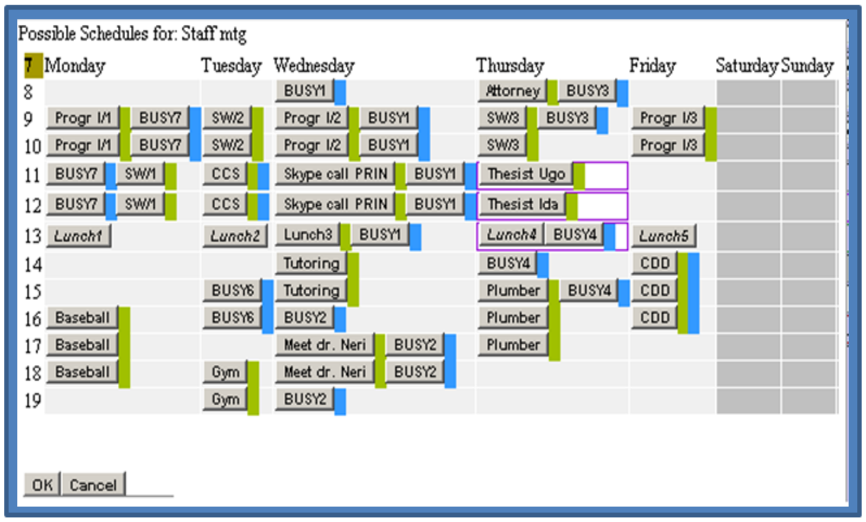
Suppose that Gianluca has to schedule a 2-hours Staff meeting with Prof. Rossi on Thursday. The event can start at 8.00 and must finish by 16.00 (deadline). In order to accommodate this meeting, the current calendar has to be revised. Prof. Rossi is available starting from 10.00 until 13.00. However, Gianluca is busy at that time because he teaches SW from 9.00 to 11.00 and then he meets two students (Thesist Ugo / Ida). The SW lesson cannot be moved. Thus, the only solution is to move Thesist Ugo and Thesist Ida to suitable, alternative times. As calendar revision is a frequent daily activity, an automatic support to this type of activity is crucial to save time.

2.2 Representation of Calendars and Calendar Items

In MARA a calendar is modeled as the collection of items belonging to its owner plus its calendar sharing information. An item belongs to the calendars of all of its participants and cross-calendar scheduling is based on the analysis of the temporal constraints associated to the pool of items having such actors as participants. Calendar items are represented as objects having the following features:

- *name* and *internal identifier*;
- *duration* - number of hours devoted to the item;
- *earliest start time* - earliest time point for scheduling the item;
- *schedule* - current temporal allocation of the item within its feasible allocation interval (depending on temporal constraints);
- *deadline* for its completion/end (if any);
- list of *participants* - this identifies the calendars to which the item belongs;
- *priority* (low, medium, high) - criticity of the commitment;
- *description* - description of the topic of the item.

Moreover, temporal order relations can be imposed on calendar items to specify that they cannot overlap (e.g., the user cannot attend two meetings at the same time) or have to be scheduled one after the other. Notice that we selected a single representation format for all types of items (tasks, meetings, to-dos, etc.), leaving most fields of the descriptor as optional. However, simple items can be described as trivial tasks lacking the specification of the actions to be performed, deadlines, and so forth.


Possible Schedules for: Staff mtg

7	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
8			BUSY1	Attorney BUSY3			
9	Progr I/1 BUSY7	SW/2	Progr I/2 BUSY1	SW/3 BUSY3	Progr I/3		
10	Progr I/1 BUSY7	SW/2	Progr I/2 BUSY1	SW/3	Progr I/3		
11	BUSY7 SW/1	CCS	Skype call PRIN BUSY1	Thesist Ugo			
12	BUSY7 SW/1	CCS	Skype call PRIN BUSY1	Thesist Ida			
13	Lunch1	Lunch2	Lunch3 BUSY1	Lunch4 BUSY4	Lunch5		
14			Tutoring	BUSY4	CDD		
15		BUSY6	Tutoring	Plumber BUSY4	CDD		
16	Baseball	BUSY6	BUSY2	Plumber	CDD		
17	Baseball		Meet dr. Neri BUSY2	Plumber			
18	Baseball	Gym	Meet dr. Neri BUSY2				
19		Gym	BUSY2				

Fig. 2. Pop-up window suggesting the time slots where the new task could be placed.

2.3 Mixed-Initiative Scheduling Support

MARA automatizes the temporal reasoning steps behind calendar management by analyzing the commitments of actors and searching for safe schedules which answer the user's needs and satisfy the temporal constraints of all calendar items. Moreover, it enables the user to steer the scheduling activity by making her/him aware of the available options and of their consequences on the schedules of the involved people.

While the user adds new items or revises the existing ones, the system checks whether the temporal constraints of the involved calendar items are satisfied. If any conflicts occur, it helps the user incrementally explore the solution space in order to quickly evaluate the available options. Specifically:

– If the user's action causes any conflicts the system suggests to click the "Where can I place the task?" (WPT) button in order to ask for help. Notice that this button is available to enable the user to receive the system's suggestions at any time, regardless of the occurrence of conflicts.
– Each time the user clicks the WPT button, MARA generates a pop-up window which presents the scheduling options; see Figure 2. The window shows the feasible time intervals (white rectangles) where the item could be placed, possibly after having moved other items in order to solve the existing temporal conflicts. In order to help the user to assess the situation, each feasible interval has a colored frame which summarizes the criticity of the changes to the involved calendars. A green frame means that all participants are available in the time slot; a violet one denotes that somebody has a low/medium-priority commitment at that time. A red one denotes the presence of high-priority commitments. Figure 2 shows the pop-up window generated in our example and suggests to place Staff meeting on Thursday

between 11.00 and 13.00 by displaying the interval in a violet frame because, at that time, Gianluca and/or Prof. Rossi have low or medium-priority commitments.

– By moving the mouse over an interval the user can view the names of the actors whose previous commitments should be moved to accommodate the item.
– If the user selects a suggestion, the system presents a few possible calendar revisions. The user can accept a suggestion (OK button) or go back to the available options (Cancel). In our example the user chooses to place Staff meeting at 12.00. Then, the system proposes two options: (a) leave Thesist Ugo at 11.00 and move Thesist Ida at 14.00; (b) move Thesist Ugo and Thesist Ida after the Staff meeting, respectively at 14.00 and 15.00, and postpone the Plumber at 16.00.
– Given the user's decision, the system informs the affected actors and applies the changes only if all of the actors accept them.[1]

2.4 Software Components

MARA stores the information about calendars, items and actors as lists of objects and exploits two modules for time reasoning purposes:

1. The *scheduling module* (see [2]) checks the consistency of the temporal constraints associated to calendar items to verify whether there is at least one safe schedule. Moreover, it implements the scheduling policies offered by the system. Currently, two policies are offered in order to schedule early, or urgent tasks first; however, other ones could be added.
2. The *Interval-based TEmporal Reasoner (ITER)* (see Sections 3.2 and 3.3) supports the revision of calendars to add or move items by identifying the feasible intervals where they could be allocated and the impact on the schedules of the involved actors. Moreover it generates the conservative schedules for each feasible interval.

The core of the smart calendar manager manages the mixed-initiative scheduling model as follows: while the user revises a calendar, the core invokes the scheduling module to check whether the user's actions violate any temporal constraints and, in case they do, it notifies the user. Each time a conflict occurs, the core invokes ITER specifying the current schedule and the problematic item in order to retrieve the feasible time intervals for safely allocating the item. For each interval selected by the user, it invokes ITER to retrieve the conservative calendar revisions to choose from.

MARA is implemented as a Java Web application. The user interface is based on the Google Web Toolkit [3]. The scheduling module exploits the JaCoP Constraint Solver [4] and ITER is developed in Perl using the Graph.pm extension module [5] for representing and manipulating STNs. The minimization of the STNs is performed by using the Floyd-Warshall algorithm included in Graph.pm. The Java Web application invokes ITER as a local REST service via HTTP.

[1] The investigation of a suitable negotiation protocol is part of our future work.

3 Temporal Reasoning

3.1 Constraint-based Representation of Calendar Items

The scheduling module and ITER exploit reasoning techniques based on Temporal Constraint Satisfaction Problems (TCSP) [6]. TCSPs are a class of Constraint Satisfaction Problems (CSPs) tailored to the representation of temporal constraints. A TCSP involves a set of variables $X_1, \ldots, X_n$ with continuous domains representing time points.

Specifically, ITER employs a subclass of TCSPs, the Simple Temporal Problems (STPs) [6], where all of the constraints are binary and they do not contain any disjunctions, i.e., they have the following format: ($a \leq X_j - X_i \leq b$). This class of problems can be represented as a graph named Simple Temporal Network (STN), whose consistency can be checked in polynomial time [7]. The *minimization* of an STN (computing for each pair of variables X_i, X_j an interval $[a_{min}, b_{min}]$ which guarantees the existence of a global solution for the STN) can be done in polynomial time, as well.

In MARA, a calendar is a collection of calendar items, possibly having multiple participants, and the scheduling of a set of calendars is done by reasoning on the joint set of constraints associated to their items. Each item M is represented by means of:

– Two numeric variables M_s and M_e, representing the start and end time of M in a given schedule (M_s corresponds to the “schedule” feature of the item). For simplicity, we assume that the value of a variable M_s (resp. M_e) is the number of one-hour slots between Monday 8.00 and the start (respectively the end) of M.
– The temporal constraints on M_s and M_e needed to schedule M consistently with its earliest start time, duration and deadline.
– The temporal constraints on M_s and M_e needed to impose precedence relations and non-overlap temporal constraints with respect to other calendar items.

For instance, let’s consider Staff meeting (SM) in Figure 1:

– The *earliest start time*, Thursday at 8.00, is expressed as $SM_s \geq 36$ because in the calendar there are 36 one-hour slots between Monday 8.00 and Thursday 8.00.
– The *deadline*, Thursday 16.00, is expressed as $SM_e \leq 44$.
– The *duration*, 2 hours, is $SM_e - SM_s = 2$ as the item takes 2 time slots.
– A *precedence* relation among calendar items, e.g., the fact that Staff meeting must take place after Skype call PRIN (SC), is expressed as $SM_s - SC_e \geq 0$.
– The fact that, e.g., Staff meeting cannot overlap SW/3 ($S3$) is expressed as $S3_s - SM_e \geq 0 \vee SM_s - S3_e \geq 0$; i.e., either SM ends before $S3$ starts or vice versa.[2]

3.2 ITER: Computing the Feasible Intervals for an Item M

Given the temporal constraints of the calendar and an item M to be added, ITER:

– Searches for feasible intervals for allocating M. For each interval, it computes the criticity on the basis of the priorities of the items which should be moved to allocate M in the corresponding time window.

[2] Non-overlap constraints can be expressed only in general TCSPs. Therefore, they are handled by the scheduling module.

ALGORITHM 1: Feasible intervals for starting a new calendar item M for a specific user U_i.

```
input:
    new item $M$
    other $U_i$ items in current schedule order $(T_{i,1}, \dots, T_{i,ki})$
    STN $\mathcal{N}_i$ (deadlines, durations, precedences for $M, T_{i,1}, \dots, T_{i,ki}$)
    $\mathcal{S}_i$ current schedule for $T_{i,1}, \dots, T_{i,ki}$
1 $(\mathcal{I}_i^{\perp}, \mathcal{I}_i^{\mathtt{L}}, \mathcal{I}_i^{\mathtt{M}}, \mathcal{I}_i^{\mathtt{H}}) \leftarrow$ Intervals($M$, $(T_{i,1}, \dots, T_{i,ki})$, $\mathcal{N}_i$, $\mathcal{S}_i$);
2 $\mathcal{I}_i \leftarrow$ DComp($\mathcal{I}_i^{\perp}, \mathcal{I}_i^{\mathtt{L}}, \mathcal{I}_i^{\mathtt{M}}, \mathcal{I}_i^{\mathtt{H}}$);
3 return $\mathcal{I}_i$
```

– Generates the conservative schedules for each feasible interval selected by the user. Intuitively, the conservative calendar revisions are enforced by adding temporal constraints which impose to maintain the relative order of existing items in the schedules.

For simplicity, the system currently only considers feasible intervals which do not require to move any already scheduled items involving multiple users. However, it can be extended to deal with a more general case with limited effort.

We introduce a notation in order to tag feasible intervals with information about their criticity. Each computed interval $I_{M,i}$ has an associated *label*:

$$l(I_{M,i}) = \{U_1^{C1}, \dots, U_m^{Cm}\}$$

where $U_1, \dots, U_m$ are the involved users and the C_i superscripts denote the criticity of the interval for each actor U_i; i.e., the impact of allocating M in the interval on U_i's calendar, in terms of revisions. Specifically, $C_i \in (\perp, \mathtt{L}, \mathtt{M}, \mathtt{H})$, where:

– $\perp$ means that the schedule of U_i is not affected by M;
– $\mathtt{L}$: only low-importance commitments of U_i have to be shifted to allocate M;
– $\mathtt{M}$: only medium- and low-importance commitments of U_i have to be shifted;
– $\mathtt{H}$: at least one high-importance commitment of U_i has to be shifted;

The computation of the feasible intervals $\mathcal{I}_M$ for M is split in two steps:

1. Computing the feasible intervals $\mathcal{I}_i = (I_{i,1}, \dots, I_{i,ni})$ for each user U_i. Each interval $I_{i,j}$ is assigned a label $l(I_{i,j}) = U_i^{Cj}$ with $C_j \in (\perp, \mathtt{L}, \mathtt{M}, \mathtt{H})$; the label specifies how critical is to allocate M in that interval, given the temporal constraints of U_i's commitments.
2. Computing the joint feasible intervals $\mathcal{I}_M$ and their labels from user intervals $\mathcal{I}_i$ in order to propose a small set of options to be considered for scheduling decisions.

Computing the User Intervals Algorithm 1 (see the corresponding figure) concerns a single user U_i and implements the first step. It takes as inputs: (i) the new item M to be allocated; (ii) U_i's other items $(T_{i,1}, \dots, T_{i,ki})$ in the order in which they appear in the current schedule; (iii) an STN $\mathcal{N}_i$ encoding the deadline, duration and precedence constraints for $T_{i,1}, \dots, T_{i,ki}$ and M; (iv) the current scheduled times $\mathcal{S}_i$ for $T_{i,1}, \dots, T_{i,ki}$.

In line 1, the algorithm computes four sequences of intervals $\mathcal{I}_i^{\perp}$, $\mathcal{I}_i^{\mathtt{L}}$, $\mathcal{I}_i^{\mathtt{M}}$, $\mathcal{I}_i^{\mathtt{H}}$ by invoking procedure *Intervals*, described below. Each sequence $\mathcal{I}_i^C$ contains $(k_i + 1)$ intervals, one for each position where M can be placed in the order of the existing items $T_{i,1}, \dots, T_{i,ki}$. For each position j, interval $I_{i,j}^C$ represents the set of time points t such that starting M at time t requires to modify U_i's schedule by shifting items of importance C or lower.

The intervals in sequences $\mathcal{I}_i^{\perp}$, $\mathcal{I}_i^{\mathtt{L}}$, $\mathcal{I}_i^{\mathtt{M}}$, $\mathcal{I}_i^{\mathtt{H}}$ can overlap both within the same sequence and among different sequences. Thus, in line 2 the algorithm invokes procedure *DComp* to merge $\mathcal{I}_i^{\perp}$, $\mathcal{I}_i^{\mathtt{L}}$, $\mathcal{I}_i^{\mathtt{M}}$, $\mathcal{I}_i^{\mathtt{H}}$ into a single sequence $\mathcal{I}_i$ of labeled and ordered disjoint intervals. Each overlapping portion is labeled with the lowest (i.e., best) criticality class to which it belongs.

Procedure *Intervals* considers each possible positioning of M in the sequence of existing items $T_{i,1}, \dots, T_{i,ki}$ involving U_i. In this way a total order Π is determined among all the items, including M, and can be asserted as a set of precedence constraints into the STN. Then, *Intervals* considers each class of items $C \in (\mathtt{H}, \mathtt{M}, \mathtt{L}, \perp)$ imposing again the associated constraints into the STN. Each resulting STN is then minimized, yielding a feasible interval $I_{i,j}^C = [min, max]$ for the start of M. As said above, for each position j of M, such an interval represents the set of time points t such that starting M at time t requires to shift items having importance C or lower in U_i's schedule. Intervals $I_{i,j}^C$ are added to their corresponding sequences $\mathcal{I}_i^C$ and, after all the positions have been considered, such sequences are returned.

Procedure *DComp* receives sequences $\mathcal{I}_i^{\perp}, \mathcal{I}_i^L, \mathcal{I}_i^M, \mathcal{I}_i^H$ and composes them into a single sequence $\mathcal{I}_i$ of ordered, labeled, disjoint intervals. Intuitively, *DComp* orders the start and end time points of each of the input intervals storing them in a list $\mathcal{T}$. Then, it scans the list generating output intervals $I_{i,j} = [t_s, t_e]$ where:

- t_s and t_e are time points in list $\mathcal{T}$, such that
- starting the new meeting M at each point in $[t_s, t_e]$ has on U_i's current schedule a minimum impact C (e.g., M for medium)

The label of such an output interval $I_{i,j}$ will then be $l(I_{i,j}) = U_i^C$.

Computing the Joint Intervals For each user U_i involved in M, Algorithm 1 computes a sequence of ordered, disjoint intervals $\mathcal{I}_i$ such that each interval $I_{i,j}$ has a label $l(I_{i,j})$ which takes values in U_i^C, $C \in (\perp, \mathtt{L}, \mathtt{M}, \mathtt{H})$.

Given that the involved users are $\{U_1, \dots, U_m\}$, ITER computes a single sequence of ordered, disjoint intervals $\mathcal{I}_M = (I_{M,1}, \dots, I_{M,n})$ such that each element $I_{M,j}$ of $\mathcal{I}_M$ represents a jointly feasible interval for starting M. Interval $I_{M,j}$ is labeled based on the labels of the user intervals from which it is derived. Given $\mathcal{I}_1, \dots, \mathcal{I}_m$, the sequence of jointly feasible intervals $\mathcal{I}_M$ satisfies the following conditions:

- Two time points t, t' belong to an interval $I_{M,j}$ iff for each involved user U_i they belong to the same user interval $I_{i,j_i} \in \mathcal{I}_i$.
- The label $l(I_{M,j})$ associated with interval $I_{M,j}$ is given by $\bigcup_i l(I_{i,j_i})$.

For example, if U_1 has a feasible interval $I_1 = [39, 42]$ with label U_1^M and U_2 has a feasible interval $I_2 = [38, 40]$ with label U_2^L, the joint interval $[39, 40]$ has label $\{U_1^M, U_2^L\}$.

The computation of $\mathcal{I}_M$ is performed by a procedure *JComp* (joint composition) analogous to *DComp*. The procedure receives the sequences of intervals $\mathcal{I}_i$ for all users U_i and produces the single sequence $\mathcal{I}_M$ of jointly feasible intervals.

3.3 ITER: Computing Revisions to a Calendar

We briefly describe the computation of revisions to shared calendars, because it only requires to run again the *Intervals* procedure used for computing the feasible intervals.

Let us suppose that one of the feasible intervals returned to the user was $I = [t_s, t_e]$ with label $l(I)$, and that the user decided to place the new meeting M at time $t \in I$. In order to compute the calendar revision, *Intervals* has to be invoked with the additional constraint that M starts at time point t and that this allocation has impacts given by $l(I)$ on the users calendars.

For each user, *Intervals* could find more than one position of the new meeting M in the task ordering that satisfies both the start time of M and the impacts $l(I)$. ITER places M in the first available position. For stability purposes, each existing task that must be moved as a consequence of adding M is started at a time as close as possible as its previously scheduled start time.

4 Tests with Users

We conducted a preliminary test to evaluate the impact on users' performance and experience of the conservative, mixed-initiative scheduling support offered by MARA. We had two research questions:

1. *Does the mixed-initiative scheduling support offered by MARA improve people's performance during the allocation of items in a shared calendar with respect to the kind of support offered by a standard calendar manager?*
2. *How is this conservative, mixed-initiative scheduling support perceived by users?*

18 volunteers were involved as participants in this experiment (10 men and 8 women). They had a median age of 40 (average = 44, minimum 17, maximum 81) and represented different categories of students and working people. They had heterogeneous backgrounds (e.g., computer science, mathematics, medicine, humanities). All participants but two were acquainted with on-line calendars (mostly Google Calendar). Participants performed the test for free, without any reward.

4.1 The Experiment

The experiment had a single-factor, within-subjects design. Two treatments were applied: in the experimental one participants used MARA and they could click the "Where can I place the task?" (WPT) button to receive suggestions for allocating items; moreover, they could preview the suggested calendar revisions by visualizing the schedules proposed by the system. In the base-case treatment, people used the same system but the WPT button and the system's suggestions (pop-up windows displaying feasible allocation intervals and possible schedules) were disabled; therefore the user had to identify possible calendar revisions and to manually apply them. Each treatment condition was considered as an independent variable. Participants' performance was considered as a dependent variable and was calculated considering the time needed to complete the task. Each participant received both treatments but their order was counterbalanced to minimize the effects of practice and fatigue.

In the experimental task each participant could access a one-week calendar shared with four people and including items involving multiple actors. Figure 1 shows the events scheduled in the calendar. The participant was asked to add a new meeting involving her/himself and another actor. The change raised temporal conflicts with existing commitments. Before starting the experimental task, participants were briefed about the scenario and they could use the system for a few minutes to get acquainted with it. Then they read the description of the scenario and the instructions describing the operations to be performed, which were available as a reference during the whole experiment. Sessions were clocked by the experimenter, who annotated all interesting actions and comments by participants while sitting at some distance from them. At the end, they were asked to answer a questionnaire for evaluating their experience in the two treatment conditions. In that questionnaire they rated the system's (i) efficacy, (ii) efficiency and (iii) usage simplicity on a 7-point Likert scale. Moreover, they rated in the same scale the usefulness of the mixed-initiative support regarding (iv) the visualization of synthetic information about the feasible intervals for allocating calendar items, (v) the preview of the revision suggestions and (vi) the conservative nature of such suggestions.

Table 1. Participants' evaluation of our smart calendar manager: mean and standard deviation values for treatment (TREAT) and base-case (BASE) conditions. Significance of results.

#	TREAT mean values	TREAT std. dev.	BASE mean values	BASE std.dev.	TREAT-BASE t	TREAT-BASE p
Efficacy	5,421	1,387	4,526	1,172	2,623	0,017
Efficiency	5,315	1,416	4,315	1,293	2,243	0,038
Simplicity	4,789	1,397	4,210	1,315	1,568	0,134
Intervals	5,736	1,484	-	-	-	0,000
Preview	5,421	1,773	-	-	-	0,000
Conserv.	5,526	1,218	-	-	-	0,000

4.2 Results

The mean execution times for the base-case and experimental treatments are 238,05 and 118,55 seconds, respectively. This shows that people take much less time to find a good revision of their calendar when supported by the system. A Paired-Samples T-test with significance level $\alpha = 0.05$ showed a significant effect on execution times.

In the final questionnaire participants rated the system's (i) efficacy, (ii) efficiency and (iii) usage simplicity on a 7-point Likert scale (from 1, the worst value, to 7, the best one). Moreover, they rated in the same scale the usefulness of the mixed-initiative support regarding (iv) the visualization of synthetic information about the feasible intervals for allocating items, (v) the preview of the revision suggestions and (vi) the conservative nature of such suggestions.

The higher portion of Table 1 reports mean and standard deviation values for the efficacy, efficiency and simplicity parameters with significance values. It can be seen that participants positively evaluated the efficacy and efficiency of the system support while there is no significant data about usage simplicity.

The lower portion of the table reports the mean values, standard deviations and significance values (obtained by means of a one-sided T-test with $\alpha = 0.05$) concerning parameters (iv), (v) and (vi).

The comments provided by participants show that they strongly appreciated the awareness support provided by the visualization of the feasible intervals because it helped them to quickly understand the pros and cons of the available options. Overall, these results suggest a positive impact of the mixed-initiative scheduling support offered by MARA on user performance and experience.

5 Related Work

The idea of developing mixed-initiative approaches for calendar management was first developed in the 90's (e.g., see [**?**]). However, since then, little work has been carried out on that topic.

Most calendar managers (e.g., Google Calendar) only support the joint visualization of a set of shared calendars. Some recent ones (e.g., Google Calendar Smart Rescheduler [9]) analyze the estimated duration and temporal constraints of the items to be scheduled in order to identify the available time slots where they could be allocated. However, they cannot suggest any schedule revisions for addressing temporal conflicts.

Many task managers such as Things [10] and Standss Smart Schedules for Outlook [11] manage tasks, deadlines and task dependencies but they have no scheduling capabilities either. Other ones are much more powerful but they are too complex for everyday activity management. For instance, the scheduler described in [12], based on a constraint optimization model, supports some flexible task scheduling policies but requires the specification of a fairly large amount of information for each task to be handled. Moreover, it only deals with single-user tasks.

Opportunistic schedulers, typically based on planning technology, synchronously guide the user in the execution of activities according to the pending goals to be achieved. However, they do not present an overview of long-term schedules. Moreover, they do not deal with shared activities and they are not mixed-initiative; they basically suggest opportunities of action, which the user may accept or ignore. For instance, see [13].

To the best of our knowledge, the only calendar manager which supports mixed-initiative scheduling is PTIME [1], which learns the user's preference and generates personalized scheduling options to choose from. Our mixed-initiative interaction model differs from the PTIME one because it puts the user in more control during the scheduling activity: our model enables the user to incrementally explore different scheduling hypotheses, starting from the possible intervals during which calendar items could be placed, and selecting the particular spots to be used on the basis of their impact on the calendars of the involved people.

6 Conclusions

We described the MARA (Mixed-initiative cAlendaR mAnager), a prototype system supporting the management of online shared calendars including multi-user activities. The main features of MARA are (i) a mixed-initiative scheduling support during calendar management, and (ii) the suggestion of conservative schedule revisions maintaining stable calendars. Such features are based on the adoption of Temporal Constraint Satisfaction techniques. A preliminary test with users provided encouraging results on the efficacy and usefulness of MARA's conservative, mixed-initiative calendar management support. Our future work concerns the development of MARA's mobile user interface, the improvement of the negotiation protocol to agree on calendar revisions and a further enhancement of the user awareness support.

References

1. Berry, P., Gervasio, M., Peintner, B., Yorke-Smith, N.: PTIME: Personalized assistance for calendaring. ACM Transactions on Intelligent Systems **2**(4) (2011) 40:1–22
2. Ardissono, L., Segnan, G.P.M., Torta, G.: Mixed-initiative management of online calendars. In: LNBIP, Web Information Systems and Technologies, Springer (2013) 167–182
3. Google: Google Web Toolkit, http://code.google.com/intl/it-IT/webtoolkit/ (2010)
4. JaCoP: JaCoP - Java Constraint Programming solver, http://www.jacop.eu/ (2011)
5. Hietaniemi, J.: Graph-0.94, http://search.cpan.org/ jhi/Graph-0.94/ (2010)
6. Dechter, R., Meiri, I., Pearl, J.: Temporal constraint networks. Art. Intell. **49** (1991) 61–95
7. Planken, L., de Weerdt, M., van der Krogt, R.: Computing all-pairs shortest paths by leveraging low treewidth. In: In Proceedings of the 21st Int. Conf. on Automated Planning and Scheduling (ICAPS-11). (2011)
8. Cesta, A., D'aloisi, D., Brancaleonii, R.: Considering the user in mixed-initiative meeting management. In: Proc. 2nd ERCIM Workshop on User Interfaces for All, Prague (1996)
9. Marmaros, D.: Smart rescheduler in google calendar labs, http://gmailblog.blogspot.it/2010/03/smart-rescheduler-in-google-calendar-labs.html (2010)
10. Cultured Code: Things Mac, http://culturedcode.com/things/ (2011)
11. Standss: Standss smart schedules for outlook, http://www.standss.com/smartschedules/default.asp (2012)
12. Refanidis, I., Yorke-Smith, N.: A constraint-based approach to scheduling an individual's activities. ACM Transactions on Intelligent Systems **1**(2) (2010) 12:1–32
13. Horvitz, E., Subramani, M.: Mobile opportunistic planning: methods and models. In: LNAI n. 4511: Proc. 11th Int. Conf. on User Modeling, Corfu, Greece (2007) 228–237