On Scheduling Events and Tasks by an Intelligent Calendar Assistant

Ioannis Refanidis[1,2] and Neil Yorke-Smith[2]

[1]University of Macedonia, Dept. of Applied Informatics, Egnatia str. 156, 54006, Thessaloniki, Greece
[2] Artificial Intelligence Center, SRI International, Menlo Park, CA 94025, USA.
yrefanid@uom.gr, nysmith@ai.sri.com

Abstract

In the last decade various research efforts and commercial offerings have sought to provide levels of automated assistance with time management. Besides basic calendaring, collaboration, and communication, many of these efforts concern event negotiation or scheduling, while others concern task management and monitoring. When automated scheduling is employed, most often events and tasks are treated separately, with the latter being kept out of the user's calendar. One consequence can be missing deadlines or even not accomplishing some tasks. Based on experience in developing and deploying an automated task management system and an event scheduling system, this paper sets the requirements for an intelligent calendar assistant that treats tasks and events in a unified way, schedules all of them within the user's calendar, and reschedules them whenever needed. We outline a set of attributes used to represent the overall combined scheduling problem, together with a set of criteria used to represent users' preferences and evaluate alternative schedules.

Introduction

Electronic calendar applications are these days ubiquitous. Common commercial applications, such as Apple iCal, Google Calendar, and Microsoft Outlook, embed some artificial intelligence techniques, such as natural language parsing, but are primitive in comparison with the state of the art in scheduling algorithms. Among these applications, some provide assistance with the related aspect of task (or to-do) management, as do dedicated offerings, such as Chandler, OmniFocus and rememberthemilk.com.[1]

Time and task management are co-dependent. Indeed, some applications, such as Chandler, deliberately blur any distinction between email message, event, and task. The distinction between events and tasks is rather conceptual. People tend to consider as event whatever has a specific time window and location, probably engaging other people too (e.g., a doctor's appointment at 14:00 on Tuesday); on the other hand, tasks usually are characterized by a deadline (e.g., write a paper). Events are usually placed directly into the user's calendar, whereas tasks are usually kept in to-do lists. This distinction is clear in many commercial applications, like Microsoft Outlook or Google Calendar, with the former giving the option to transform a task to an event by dropping it in the calendar.

Research in event scheduling has mainly concentrated on coordinating information and arranging meetings between groups of people. In the research community, several approaches have developed to schedule individual or group events, potentially taking into account user preferences, in order to develop intelligent calendaring assistants (e.g., Mitchell et al. 1994; Sen and Durfee 1994; Jennings and Jackson 1995; Modi et al. 2004; Berry et al. 2006, 2009). On the other hand, task management research efforts have concentrated on facilitating task assignment, execution monitoring, and raising reminders (e.g., Bellotti et al. 2004; Conley and Carpenter 2007) or, in case of scheduling tasks into a user's calendar, completely ignore meeting scheduling (Refanidis and Alexiadis 2008). However, since both events and tasks require the same resource—the user's time—treating them separately may result in poor overall utilization and unnecessary user effort in overall time management.

In this paper we propose to treat events and tasks in a unified way. Our proposal is based on our experience from working on systems like SELFPLANNER (Refanidis and Alexiadis 2008) and *Emma* (Berry et al. 2008). We outline a rich set of attributes shared by events and tasks, giving motivational examples. This set of attributes results from thorough study of the common situations that arise in practice. We also sketch a hierarchical preference model that could be used (in entirety or in parts) to evaluate alternative schedules. Finally, we discuss some engineering issues that concern the development of an intelligent calendar assistant based on (aspects of) the proposed model. The goal of this paper is not to present a rigorous theoretical model; it is rather to emphasize the multiple aspects and inherent complexities of the problem of organizing a user's time and, thus, provide with ideas for subsequent formulations and implementations.

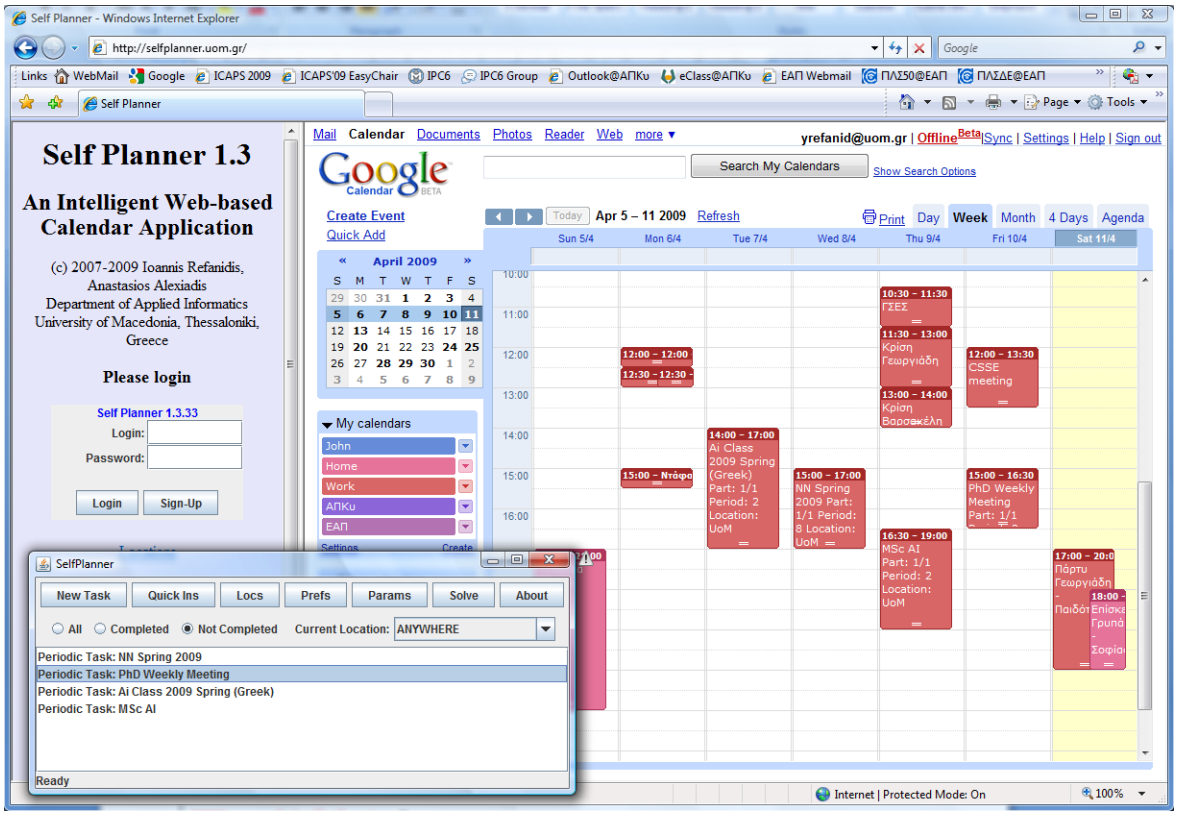

Figure 1. Screenshot of SELFPLANNER (http://selfplanner.uom.gr)

Our Previous Work

We outline two deployed systems, *Emma* and SELFPLANNER, focusing on their complementary differences. The features of an envisioned new scheduling system, inspired by the best features of the existing systems, are presented more formally in the subsequent sections.

SELFPLANNER aims at helping the user to schedule her own individual tasks. Each task is characterized by its duration and temporal domain (among several other attributes). SELFPLANNER tries to accommodate the tasks within the user's calendar, taking into account a variety of constraints. SELFPLANNER emphasizes the use of automated scheduling algorithms (Refanidis 2007). Figure 1 displays a screenshot of the system.

Emma aims at helping the user to schedule events (e.g., group meetings), while taking into account learnt user preferences. Upon a meeting request that involves other participants, Emma reads the calendars of all participants and generates a set of potential schedules for the meeting. It ranks each schedule option by taking into account the participants' preferences (learnt or elicited) and their relative importance for the organizer. Emma thus adopts a mixed-initiative paradigm. Having scheduled an event, Emma generally attempts to not reschedule it. Figure 2 gives a screenshot of the system.

Since the two approaches are complementary in several ways, combining their best features in a single intelligent calendar assistant would be worthy. In the next paragraphs we comparatively present the two systems, focusing on their complementarities.

Overlapping events and tasks: Emma allows overlapping events. In particular, it is up to the user to decide whether she wants overlapping events or not. Having overlapping events may have one of the following meanings for the user:

- The user will attend all events concurrently (for example attending a physical meeting while participating in a videoconference).
- The user will decide at a later time which event(s) to attend.
- The user has initially blocked a time period for a specific event (e.g., attending a conference) and then she added new events describing detailed activities within the main event (e.g., conference sections, banquet etc).
- Similar to the previous case, the user has initially reserved a block of time for just in case (e.g., it might be the case that this day something is going to happen). At a later time, more specific events are added into the calendar, overlapping with the initial event.
- The user has blocked some period of time, just to

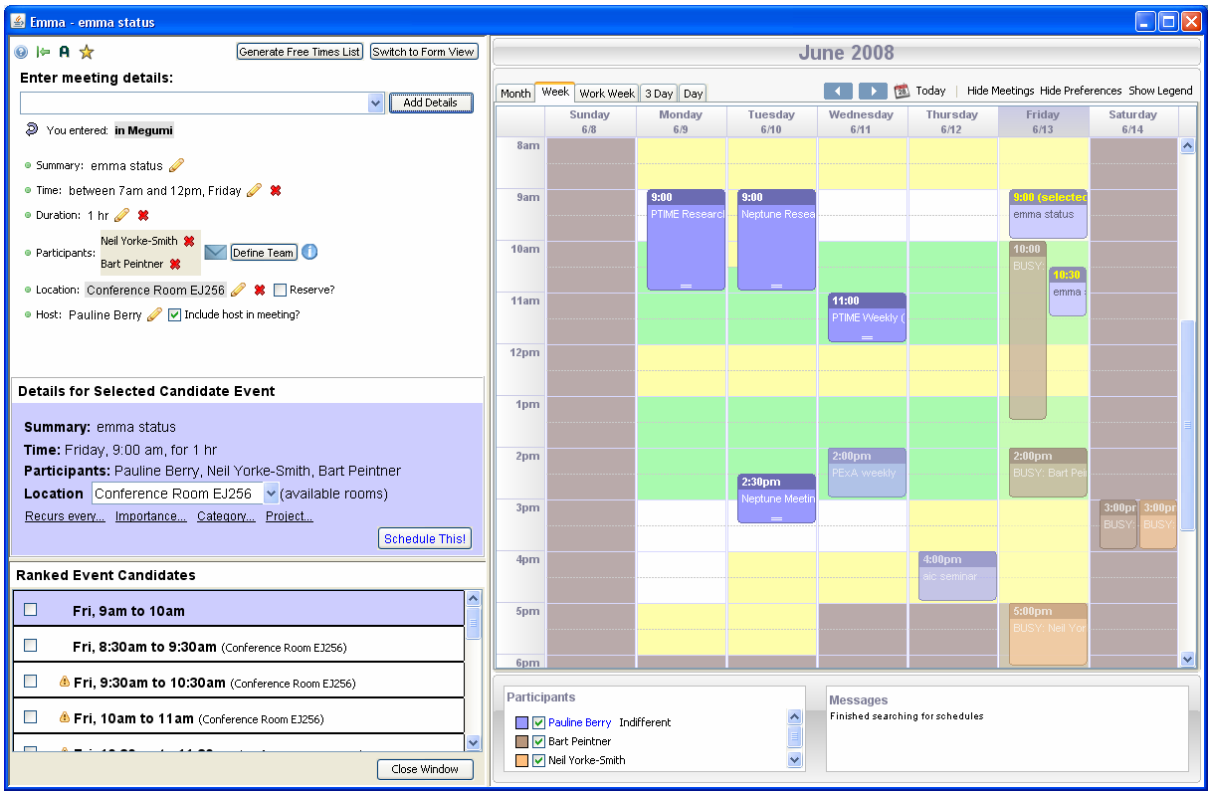

Figure 2. Screenshot of *Emma*

prevent other people (including himself) from putting events over this period (of course this presupposes that she gave the right to other people to insert or at least view events on her calendar). Later the user added new events overlapping the blocked out period.

- Somebody else has added an event to the user's calendar (e.g., a meeting request that has been automatically inserted into the user's calendar), that overlaps with existing events.
- The events might have duration less than the system's quantum of time. For example, in case the system's quantum of time is 30', one could schedule two events of 15' duration each in the same time slice, and both of them will be attended by the user (of course the two events should have compatible locations).

Note that Emma does not automatically schedule the events but proposes rated alternative schedules to the user who eventually decides when (and where) to schedule each event. Thus, the user decides whether to have parallel events or not.

SELFPLANNER does not support overlapping events. This is a consequence of the fact that SELFPLANNER solves a different scheduling problem where all tasks need a single resource, i.e., the user.

Locations: SELFPLANNER supports locations attached to the tasks. The system employs Google Maps service to compute temporal distances between the locations, so as to leave the necessary temporal distance between adjacent tasks scheduled at different locations. Furthermore, SELFPLANNER supports location classes denoting sets of locations. A location class can be attached to a task; in that case the scheduling problem involves the selection of a member of that class in order to schedule there the task. Note that locations in SELFPLANNER are not considered as resources, i.e., several tasks (of different users) can be scheduled at the same location concurrently.

Emma supports locations according to the iCal specification, i.e., the user can give value to a "where" field; however, no special spatial reasoning is undertaken.

Resources: Emma supports resource management w.r.t. a database of specific locations. That is, Emma considers various locations (i.e., meeting rooms and offices at a company) as being resources that are assigned to specific events. Each location is treated as a virtual user having its own calendar. However, no overlapping events are allowed to these special calendars, ensuring that a location cannot be reserved for two concurrent events.

Interruptible tasks: SELFPLANNER supports interruptible tasks. The user can define whether a task is interruptible or not and, in case it is, she can post additional constraints on the durations of the parts of the interruptible task or their temporal distances. Note that parts of interruptible events may be scheduled in different locations, provided that a class of locations has been assigned to the task.

Binary constraints: SELFPLANNER supports ordering constraints over pairs of tasks. An ordering constraint of the form A<B, where A and B are two tasks, means that B

cannot start its execution before A has finished. In case of A or/and B are interruptible, no part of B can start its execution before all parts of A have finished.

Preferences: SELFPLANNER supports a limited form of preferences. In particular, for each task the user can specify whether she wants the task to be scheduled as early/late as possible, or before/after a time point or finally whether she is indifferent of when to schedule the task.

Preferences in Emma are given or learnt. The user initially defines which hours within a week he mostly likes/dislikes to schedule events in her calendar. He also may define whether she dislikes events during lunch and what kind of reminders she prefers. Finally, she may relatively rate eight different factors that may affect her preference over an event: her general day/time preferences stated earlier, rescheduling, event start time, duration, overlaps, participants, location and others' preferences. Having created an initial profile, the system then monitors the user activities, which alternative schedule she selects when organizing an event, how she responds in event requests etc, in order to continuously refine a model that has an aspect for each one of the eight criteria.

Periodicity: SELFPLANNER supports periodic tasks, in particular daily, weekly, and monthly. A periodic task is considered as a collection of simple, non-periodic tasks. However, defining a task as periodic has several benefits for the user, since she specifies the parameters of the task (e.g., temporal domain, preferences) once only. Note that the various instances of the periodic task do not have to be scheduled in a uniform way.

Emma supports periodic events of five periods: hourly, daily, weekly, monthly and yearly. After the organizer of an event schedules the first instance of it, the rest of the instances are scheduled automatically in the same way as the first. However, it is always possible for a user to ask for rescheduling of any instance of the periodic event.

Distributed scheduling: Neither SELFPLANNER nor Emma support distributed scheduling. In case of SELFPLANNER this is expected, since this system does not support meeting scheduling. In case of Emma, scheduling is performed by the organizer, who chooses a scheduling option taking into account the various preferences, one of them being "other's preferences". The organizer has also the possibility to present to the participants a subset of the available options and ask for their specific opinions; however she makes the final decision. In any case the other participants may reject the event that has been tentatively scheduled.

User status: An interesting feature supported by few systems is to monitor the user's status, either explicitly or implicitly. Currently, SELFPLANNER implicitly infers the current location of the user from the location of the last accomplished event, and this is taken into account when scheduling the next one. Other attributes of the user could be whether she is on her car or not, whether she carries her laptop or not, what type of clothes she is wearing, etc. Having access to this type of information transforms the scheduling problem into a planning problem, where actions have preconditions and effects. In that case, the system should reason on what actions should be inserted into the user's plan in order to satisfy open goals. Planning algorithms like POP (Weld 1994) or mixed-initiative versions of them, would be best candidates to solve the resulting problem.

Multiple calendars: Both systems support multiple calendars. This means that each user can declare all calendars she utilizes, and the system collects information from all of them when scheduling a new event.

Partial Scheduling: SELFPLANNER supports partial scheduling: if no schedule can be found that accommodates all tasks, the best incomplete schedule is adopted and presented to the user. Furthermore, the user is informed of which tasks or parts of tasks were unable to schedule.

Emma does not support partial scheduling, but it always accommodates all events, since overlapping events are allowed whereas no explicit temporal domains or other hard constraints are defined. However, alternative schedules with many overlapping events or with events scheduled at abnormal times receive lower scores in the list of options presented to the user.

Problem Formulation

Before delving into the details of an event's description, we would like to discuss our view of hard and soft constraints. Organizing a user's calendar is a very ambitious task to be accepted by the user, so we expect from the system to be as flexible as possible by presenting to the user several alternative schedules and let him decide the best. Flexibility is greatly favored by soft constraints instead of hard ones. So, it is expected that the user will use as far as possible soft constraints (i.e., preferences) to describe events and tasks. However, there are cases where several value assignments to the decision variables are strongly prohibited. Here are some examples:

- The user wants to attend a lecture. This event has a very specific time and location. There is little value in scheduling this event in another time or location.
- The user is attempting to organize a meeting with another participant (a two-person meeting). In that case, the constraint that both participants are necessary for the meeting is hard; there is no point in having a meeting without one of the participants!
- A location with specific features is required for a meeting. In that case, there is no point in selecting a location with, e.g., lower capacity or without a teleconference facility.
- Physical constraints such as that the duration of an event cannot be a negative value or that the end time is equal to the sum of the start time and the duration.

In all the above examples there are some decision variables, such as the start time or the location, that have very specific acceptable values or ranges of values. There is no point in assigning to these variables out-of-the-

domain values; in such cases it is preferable not to schedule the event at all! However, even in such cases, whether an event can be omitted or not is something that the user decides. If an event is necessary (another type of hard constraint), the system should include it if possible. So, provided that a problem has been formulated with hard and soft constraints, we always prefer a schedule where all hard constraints are satisfied. If there is no such schedule, the system clearly notifies the user and asks for her intervention (i.e., changing the problem definition), probably by suggesting a minimum set of constraints that should be relaxed in order for the problem to become solvable. However, even for unsolvable problems the system might rank the alternative partial schedules based on several criteria, such as the sum of the utilities (or importance) of the accommodated tasks as well as the degree to which soft constraints were satisfied, and present them to the user in some decreasing order of preference.

Decision Variables and Hard Constraints

In the following we use the term 'event' to refer to an event or a task interchangeably. Towards a unified calendaring assistant that exploits AI scheduling, we propose attributes to include:

Duration range. The possible durations of an event, given as a lower and an upper bound. For example, an event of the form "Lunch" might have a duration between 30 and 60 minutes. It is expected that the utility function (if defined) over the possible durations will be a monotonically increasing function.

Temporal domain. The set of possible time periods when the event can be scheduled. Commonly, the temporal domain is represented as a disjunction of intervals. For example, the temporal domain of an event "Shopping" would not include late night or Sundays in central Europe.

Locations. A list of alternative locations where the user should be for the event. For example, the set of locations for the event "Shopping" might consist of the malls in the area. Each location is characterized from its geographic coordinates. Travelling times between locations should be taken into account by an intelligent calendar assistant when scheduling events in space and time. An "anywhere" location is also foreseen.

Reserve Location. A Boolean flag indicating whether the selected location for the event, e.g., a meeting room, should be reserved. Reservation supposes a Location Manager that answers availability queries and performs reservations. Reserving locations may impose extra temporal constraints, since the scheduled location must be available during the schedule time.

Utility. Each event is characterized by its utility. The simplest model assumes a single, constant and known positive utility value. More complex models would have the utility depending on, for instance, the duration of the event, its location, the number of participants, and so on. Utilities of events are particularly important in the case of overconstrained problems.

Utilization. A number between 0% and 100% defining whether an event requires all user's attention/effort (100%) or part of it (less than 100%). Several events of the latter type could be scheduled in parallel, provided that they are scheduled in compatible locations.

One further possibility is to suppose utilization and duration as inverse linear dependant. For example, an event's duration might be 2 hours with 100% utilization, 4 hours with 50% utilization, etc. This gives alternative scheduling options.

Optional. A Boolean flag indicating that the user is not obliged or has not yet decided to attend the event. The last two attributes, *Optional* and *Utilization*, give rise for overlapping events. Indeed, having overlapping events in a user's calendar is a common situation in practice, for various reasons described in the previous section.

Category. A category characterizes a class of events, e.g., Meetings, Teaching, Housekeeping, etc. An event may belong to several categories. We assume that an ontology of categories is provided. Pairs of categories of the ontology might be compatible or not: only events of compatible categories, provided they are optional or have proper utilization values, may overlap. Compatibility is not a reflexive relation.

The *Category* attribute can be combined with the *Utilization* and *Optional* attributes in interesting ways. For example, suppose the user wants to attend ICAPS'09 in Thessaloniki. She creates an "abstract" event with zero utilization, non-optional, with category *icaps09*, spanning the days of the conference; category *icaps09* is compatible only with itself. Next, the user might add several optional, full utilization events for various sessions of the conference, all sharing the same category. Suppose now that CP'09 runs over the same days in Lisbon, and the user has not yet decided which of the two conferences to attend; however, she will attend one for sure. This can be modelled by defining a second conference event for CP with zero utilization, optional, and category *cp09* that overlaps with the ICAPS event; category *cp09* is compatible only with itself too. Again, CP optional events may be added. As soon as the user decides which conference to attend, she only needs to define the corresponding abstract event as non-optional—constraint propagation can arrange everything else.

A subtlety with the above formulation concerns what happens as long as the two abstract events are still optional: the system may schedule any other optional event during the same period. Thus, the user needs a way to state that at least one of the two conference events should be scheduled. This can be done in two ways:

- Defining a new category, *icaps/cp09*, that is compatible only with *icaps09* and *cp09* and creating a new non-optional abstract event of zero utilization spanning over the same days. This new category should not be an ancestor of *icaps09* and *cp09* in the ontology.
- Using a binary constraint stating that at least one of the two main conference events should be scheduled.

Participants. A list of people who are invited to attend

the event. In a decentralized system's architecture, one participant is considered the organizer of the event, i.e., is the person who will decide the final schedule. For each participant, the following information may be defined:

- *Optional*. A Boolean flag indicating whether the participant is necessary for the event to occur or not.
- *Availability*. For each participant, the organizer would like to know when (and perhaps where) she is available. At its simplest, this can be done by giving the organizer access to the participant's calendar. More sophisticated negotiation and privacy protocols have been studied in Maheswaran et al. (2005).
- *MinParticipants*. The minimum number of participants that are needed in order for the event to take place (e.g., more than half of them).

Periodic. A Boolean flag denoting a repeating event. Additional attributes are needed to specify the length of the period, e.g., a week, and the number of occurrences. Each occurrence should be scheduled separately.

Interruptible. A Boolean flag denoting an event that can be scheduled in parts, e.g., writing a paper. Additional attributes are needed to specify the minimum/maximum duration of each part and the minimum/maximum temporal distance between pairs of parts. For example, the user might want at least four hours break between any two "writing the paper" sessions, but wants to finish within two months from starting working on this event.

Constraints. Binary and higher-order constraints over events should be allowed. Some obvious candidates are:

- *Temporal ordering*. Event *s before r* is interpreted as "*r* cannot start its execution before all parts of *s* have finished".
- *Temporal proximity*. A minimum/maximum interval between two events.
- *Physical proximity constraints*. A minimum/maximum distance between the locations assigned to two events.
- *M* out of *N*. At least *M* out of a set of *N* optional events should be included without overlaps.

Special interpretation is needed when applying constraints on interruptible and periodic events. For example, stating that the maximum temporal distance between two interruptible events *s* and *r* is *t* could be interpreted as that there is at least one pair of their parts s_i and r_j that have distance less than or equal to *t*. Another, perhaps less intuitive, interpretation is that any part s_i should have distance at most *t* from some part s_j and vice versa. By contrast, in case of a minimum distance constraint between events *s* and *r*, the only reasonable interpretation is that *t* is the minimum distance for any pair of parts s_i and r_j.

Similar issues arise when *s* and/or *r* are periodic. For any binary constraint over them, it has to be decided whether it concerns pairs of instances of the events (which presupposes that they have the same period), or the entire sequences of instances.

Evaluation Model

We now turn to how a system that schedules events with attributes such as those presented in the previous section, may compare candidate (partial) schedules, in order to select the best alternative or propose to the user alternative schedules in decreasing order of preference. We propose a hierarchical model consisting of five top-level criteria.

$\boldsymbol{C_1}$. *Overall utility of a schedule*, computed as the sum of the utilities of the events it involves. In case of overlapping optional events, an interesting computational problem is to select this subset of non-overlapping events that maximizes the overall utility. The problem becomes more complicated since additional preferences over the events might have been expressed, using the subsequently described criteria.

$\boldsymbol{C_2}$. *Unary preferences for an event*, aggregated over all scheduled events. This criterion concerns the utility the user perceives from the way each event *s* has been scheduled, not taking into account other events. Criterion C_2 may be composed of metrics such as the following:

p_{21}: *Temporal preference*. Suppose an event's s_i temporal domain has the form $Domain = [L_1..R_1] \cup [L_2..R_2] \cup \ldots \cup [L_k..R_k]$, where the $[L_j..R_j]$ comprise a set of non-overlapping intervals. Each time slot $[t..t+1] \in Domain$, denoted also simply with *t*, might get a utility value $u_i^{\text{TIME}}(t)$. Suppose now that $Schedule_i$ is the collection of those time slots that s_i occupies according to the current overall schedule, with $|Schedule_i|$ being the decided duration of s_i. In that case, we define the temporal preference of s_i as:

$$p_{21}(s_i) = \frac{\sum_{t \in Schedule_i} u_i^{\text{TIME}}(t)}{|\, Schedule_i \,|}$$

p_{22}: *Duration preference*. Preference over the selected duration. For example, let $Dur=[minDur,maxDur]$ is the interval of possible durations, then for each $d \in Dur$, a utility $p_{22}= u_i^{\text{DUR}}(d)$ has to be provided for s_i.

p_{23}: *Location preference*. For each event s_i, a set of locations Loc_i has been assigned to it. Each location $l \in Loc_i$ has an assigned utility $u_i^{\text{LOC}}(l)$. Suppose now that s_i has an overall scheduled duration $d \in Dur$, and has been scheduled in *N* parts with durations $d_1, d_2, \ldots, d_N$ and in locations $l_1, l_2, \ldots, l_N$ respectively, where $\sum_{j=1}^{N} d_j = d$. Then, the utility received by the current schedule of s_i concerning its location can be defined as:

$$p_{23}(s_i) = \frac{\sum_{j=1}^{N} d_j u_i^{\text{LOC}}(l_j)}{d}$$

p_{24}: *Participants*. In case participants are involved, each one of them perceives a utility from the way the current event is scheduled. Metric p_{24} considers how the organizer

of the event appreciates these utilities. The following attributes might be related to each participant:

- *Host*. This is a Boolean value indicating that this participant is the host of the event. Usually the host is also the organizer, but this might not be the case. Furthermore, however multiple hosts are allowed. Being a participant the host may affect the organizer's scheduling decisions.
- *Utility*. The value the organizer assigns to having each (optional) invited participant in the event.
- *Importance of opinion*. The weight the organizer gives to the participant's general preferences and their specific feedback on the alternative scheduling options.

p_{25}: *Parts*. In the case event s_i is interruptible, there might exist several ways to divide it into parts. The user might have preference over the way s_i has been divided, i.e., in how many parts it has been divided, what is their temporal distribution etc. So, there are several sub-metrics, over which p_{25} might be defined. The most obvious of them are the following:

- p_{251}: *Number of parts*. The user has to define her preference u_i^{PARTS} over the number of parts of s_i.
- p_{252}: *Parts durations*. The user might have a preference over the possible durations d of the various parts of event s_i. Let $u_i^{\text{PARTDUR}}(d)$ denote this preference function. Supposing the current event has been divided into N parts with durations d_1, d_2, …, d_N, the overall preference for this distribution could be defined as:

$$p_{252}(s_i) = \frac{\sum_{j=1}^{N} u_i^{\text{PARTDUR}}(d_j)}{N}$$

An instance of p_{252} could be the following:

$$p_{252}(s_i) = \frac{\sum_{j=1}^{N} \frac{1}{1+a(d_j - d^*)^2}}{N}$$

where d^* is a preferred duration and a is a parameter taking non-negative real numbers. The preferred duration d^* is provided by the user (e.g., the user prefers to work on writing a paper in parts of d^*=4 hours). Parameter a determines how deviations from this preferred duration are penalized.

Note that expressing preferences over the duration of a part is not independent from expressing preferences over the number of parts of the event. For example, it is contradicting to prefer few parts and short durations simultaneously. However, for the same number of parts, there are several ways to distribute the overall event's duration over them, so both metrics are needed.

- p_{253}: *Average distances*. The user might have a preference over the temporal distances of the various parts, e.g., she does not prefer the parts to be too close to each other. So, suppose the current event has been split into N parts, let's denote with $d_{j,j+1}$ the temporal distance between the j^{th} and the $j+1^{\text{th}}$ part of event s_i, supposing that always the $(j+1)^{\text{th}}$ part is scheduled just after the j^{th} part, for any $j \in [1..N\text{-}1]$.

  Suppose now that for any temporal distance d between parts of event s_i, the user has a preference $u_i^{\text{DIST}}(d)$ on it. Then, we could define p_{253} as:

$$p_{253}(s_i) = \frac{\sum_{j=1}^{N-1} u_i^{\text{DIST}}(d_{j,j+1})}{N}$$

- p_{254}: *Makespan*. For an interruptible event s_i, the makespan is defined as the temporal distance between the start time of the temporally first part of the event and the end time of the temporally last part of the event. The user might have a preference over the makespan of the event, i.e., prefer shorter makespan than longer (e.g., an interruptible event concerning writing a journal paper—without a deadline—should not have a makespan of 1 year, even if the temporal domain of this event is wider than a year; the user does not want to split this event in many parts that are temporally away from each other).

***C*3**. *Higher order preferences over sets of events.* This criterion aggregates expressed preferences over sets of scheduled events, such as:

p_{31}. *Ordering/proximity preference*. For two events s_i and s_j, we might have a preference of how s_i is scheduled w.r.t. s_j. This preference might concern either the order in which the two events have been scheduled, or how close to each other they have been scheduled. Obviously, there are many ways to define p_{31}, depending on whether the two events are interruptible or not and on what type of preference we are interesting in. We will opt for a general schema that can be used both for interruptible and non-interruptible events, as well as both for ordering and proximity constraints. According to this schema, p_{31} over s_i and s_j could be defined as follows:

$$p_{31}(s_i, s_j) = \frac{1}{D_i D_j} \int_{s_i} \int_{s_j} u_{i,j}^{\text{ORDERING}}(t_j - t_i) dt_j dt_i$$

In this way, $p_{31}(s_i,s_j)$ is computed as a double integral over all the various delta parts of s_i and s_j, i.e., dt_i and dt_j denote very small intervals within s_i and s_j, with t_i and t_j being their start times. D_i and D_j denote the total durations of the two events. The utility function $u_{i,j}^{\text{ORDERING}}$ might be defined in several ways, e.g., a linear or a step function for an ordering preference, a triangle or an inverse triangle function for a proximity preference. In any case (i.e. even if $u_{i,j}^{\text{ORDERING}}$ is a step function) the resulting function $p_{31}(s_i,s_j)$ is a continuous function.

p_{32}. *M out of N preference*. From a set S of N optional events s_1, s_2, …, s_N, we have preference over how many of them have been included in the schedule in a non-

conflicting way. For this purpose, an event is considered as included in the schedule if it is not overlapping with other events in a way that the user has to choose which one to attend, whereas no other constraint is violated. The user may express her preference for the various discrete values of $M \in [0..N]$, $u_S^{\text{MofN}}(M)$.

***C*₄.** *Overall preference of a schedule*. This criterion concerns attributes of the schedule as a whole, as opposed to the specific events it contains. For example:

p_{41}: *Free time*. The user might be concerned about her free time. As free we consider any time slot that has no event (optional or not) scheduled in it (including implicit traveling events). Perhaps the user might use additional criteria as to what is considered a free time, such as:

- A minimum concrete amount of free time.
- Classes of locations where the user should be in order for that time to be considered as free.
- Types of events that are compatible with the notion of free time (e.g., going to the theater is an event that could be considered free time).

Another important dimension while measuring the amount of free time concerns the horizon over which it is computed. It is expected that if we measure the amount of free time over a year, most of the time will be free since the events that will occupy this time have not yet been inserted into the user's calendar. So, more realistic horizons of, e.g., one week or a month should be considered. Perhaps, the user could measure the free time for each one of the next, e.g., four weeks and aggregate the results using higher weights for the earlier periods.

Supposing d^{FREE} is the amount of free time for some period, p_{41} can be defined as:

$$p_{41} = u^{\text{FREE}}(d^{\text{FREE}})$$

where u^{FREE} is expected to monotonically increase.

p_{42}*: Travelling*. Since most events are expected to have locations attached to them, the user's schedule will implicitly involve traveling. We expect that users would prefer as less traveling as possible. So, p_{42} is expected to be a monotonically decreasing function of the total time of travelling, the total distance of travelling and the total cost of travelling (reasoning over alternative ways of travelling would be helpful).

p_{43}: *Fragmentation*. The user might prefer her free time (equivalently her busy time) to be as compact as possible. For example, she might prefer a single block of 6 hours of free time than two 3-hour blocks. One way to measure how the free time is distributed is to divide the total free time by the number of compact pieces in which it has been distributed. For example, if the total free time is 12 hours and free time has been distributed into three parts of four hours each, then 1/3 might be a measure of the fragmentation (higher values denote less fragmentation).

However, taking into account only the number of intervals loses information about the sizes of these intervals. For example, splitting a 12 hour free time block in 4-4-4 is not the same as splitting it in 5-5-2. In that case it is difficult to judge which option is better; it depends on the user. So, to discriminate between these options too, an evaluation function over the compact free time intervals, $u^{\text{COMPACT\_FREE}}$, has to be defined. Then, for any specific fragmentation of the free time, the sum of the utilities of the distinct compact free time periods could be used to define p_{43}. A good candidate to define $u^{\text{COMPACT\_FREE}}$ could be a sigmoid function. Furthermore, to make p_{43} independent on the actual amount of free time, we could divide p_{43} with the total duration of free time:

$$p_{43} = \frac{\sum_i u^{\text{COMPACT\_FREE}}(d_i^{\text{FREE}})}{D^{\text{FREE}}}$$

where D^{FREE} is the total free time and d_i^{FREE} denotes the i^{th} compact period of free time.

p_{44}: *Balance*. This metric is similar to the fragmentation one. However, instead of measuring the distribution of the free time, *Balance* measures the distribution of the busy time. In order to evaluate p_{44}, we have to define two intervals; a period over which balance is measured and a time horizon. For example, we might be interested in balanced day schedules over the next five days. Suppose that w_i is the amount of work in the i^{th} period of interest. In that case, p_{44} could be defined as:

$$p_{44} = 1 - \frac{\text{var}(w_i)}{\overline{w_i}}$$

That is, p_{44} could be defined as the variance over the w_i values, divided by their average value. In this way, p_{44} is independent of the actual workload; it just evaluates how uniformly the workload has been distributed over the schedule. So, two schedules with different workloads but with similar distributions would have the same value on p_{44}. However, again, they would have different values in p_{41} (free time metric).

Similar to p_{41}, balance should also be measured w.r.t. specific categories of events. For example, non-business activities occupy some time slots into a user's calendar, however they should not contribute to a user's workload.

p_{45}: *Number of events*. This metric measures the number of the events included in the schedule. Contrary to p_{11}, p_{45} does not take into account the overall utility of the included events but just their number. The rationale behind this criterion is that in case two alternative schedules have the same overall utility of included events, the user might prefer the schedule with the most events, i.e., she prefers to work first on less significant but easy to accomplish events than on more significant but difficult to accomplish events. More details concerning issues such as priority and urgency of events can be found in Berry et al. (2007).

p_{46}: *Overlaps*. This metric assess the percentage of overlapping events a schedule has. A straightforward way to measure overlaps is:

$$p_{46} = \frac{D^{\text{BUSY}}}{\sum_i d_i}$$

where D^{BUSY} is the total amount of busy periods of the

schedule, i.e., periods where at least one event is scheduled and d_i is the duration of the i^{th} scheduled event (optional or not), i ranging over all scheduled events s_i. Again, specific classes of events could be ignored when computing p_{46}.

p_{47}: *Sensitivity/brittleness.* This metric measures to what extent a schedule remains valid if some events finish after their initially scheduled end-time. For each event s_i in a schedule, we can compute what delay we might afford without a need to reschedule the remaining events, apart from performing equivalent shifts to them. Let's denote this margin with m_i. Given a minimum desirable margin for each event, m_{min}, we could define p_{47} as:

$$p_{47} = \min_i \left(\frac{m_i}{m_{\min}}, 1 \right)$$

i.e., it is the delay (w.r.t. the min desirable delay) of the most critical path of the schedule (Hiatt et al. 2009).

***C*$_5$**. *One schedule compared to other schedules*. This criterion compares a schedule w.r.t. a set of reference schedules, such as the user's existing schedule or an optimized template schedule. Reference schedules can be described at any level of abstraction: they might include either very specific events or abstract ones. Reference schedules form an alternative way for the user to express preferences. Criterion C_5 may be composed of metrics such as the following:

p_{51}. *Stability/perturbation*. Given a reference schedule or template, this metric assess how much the each alternative schedule differs from the reference one. There are several ways to measure stability, depending (among others) on how the reference schedule looks like. The simplest approach is to consider the reference schedule as one containing some of the events of the current one. This situation is frequent in incremental scheduling, when a new event arrives and must be accommodated within the existing schedule, so the user might want as less perturbation to the existing schedule as possible. Measuring the perturbation between an existing schedule and a new one is however difficult to define. For example, moving an event in time or changing its location or changing the relative order or two events is usually undesirable. So, p_{51} can be analyzed in sub-metrics such as:

- p_{511}: *Average shift in time*. For all events common in both schedules, this metric assesses the average shift in time, where the average is performed over the absolute values of these shifts. A specialized approach is needed to measure average shift for interruptible events.
- p_{512}: *Average shift in space*. For all events common in both schedules, this metric assess the average distance between their new and old location. Again, a specialized approach is needed to measure average shift in space for interruptible events.
- p_{513}: *Average utility of events changed.* For all events moved compared to S' (including especially those omitted), their average utility.
- p_{514}: *Order changes*. For all events common in both schedules, this metric measures the number of pairs of events that have changed their order. Again, a specialized approach is needed to measure the number of order changes for interruptible events.

Note that the user might provide more than one reference schedule: p_{51} has to be computed for each one of them. Another dimension to be considered concerns the timing of changes: it is expected that changes in the near term schedule would be more obtrusive than changes in the long term. Metrics p_{511} through p_{514} should weigh the various changes with a function of their scheduling times, giving higher importance to near-term changes.

Aggregating the Criteria

These five criteria engender a multi-criteria evaluation of candidate schedules. We can follow prior work (Berry et al. 2006) in combining them with aggregation methods (instead of dominance relations), to result in total ordering (Keeney and Raiffa 1976). Aggregation functions can also be exploited in search as a heuristic guidance. Finally, they can be used in order to produce structurally different locally-optimal alternatives, by penalizing schedules that are similar to those already found (using, e.g., criterion C_5).

As for the specific aggregation functions, we propose two of them: a *weighted sum* when aggregating values resulted from applying the model or parts of it to different events or to different reference schedules (these values could be considered independent), and a *Choquet integral* (Labreuche and Grabisch 2003) when aggregating different criteria or sub-criteria for the same event or for the schedule as a whole (Table 1). Intelligent user interfaces and learning methods should be exploited to assess the aggregation coefficients.

Criterion	Sub-criteria	Aggregation method	Weights
Root	C_1 through C_5	Choquet integral	Elicited or learnt
C_1	Over all events	Sum (no weights)	-
C_2	Over all events	Weighted sum	Event's importance (given or learnt)
$C_2(s)$	p_{21} through p_{25}	Choquet integral	Elicited or learnt
p_{24}	Over all participants	Weighted sum	Importance of opinion (given or learnt)
p_{25}	p_{251} through p_{254}	Choquet integral	Elicited or learnt
C_3	over all higher order constraints	Weighted sum	Given or engineered
C_4	p_{41} through p_{47}	Choquet integral	Elicited or learnt
C_5	*over all reference schedules*	Weighted sum	Given
p_{51}	p_{511} *through* p_{514}	Choquet integral	Given or learnt

Table 1. Suggested methods to aggregate criteria

Conclusions and Future Work

This paper argues that organizing a user's time in an automated way is a compelling opportunity and a challenging problem. Based on our experience in the field, we presented a rich set of attributes and constraints that can be used to model this problem. We also proposed a set of criteria that could be used to evaluate alternative schedules.

Towards the implementation of an intelligent calendar assistant, we identify several engineering requirements:

- *Problem solving*. A fast, although incomplete, search algorithm could be used to produce the alternative schedules. In Refanidis (2007), Squeaky Wheel Optimization (SWO) has been used to solve the scheduling problem, using a subset of the problem formulation presented in the current paper, and employing only criterion p_{21} to evaluate the alternative schedules. Experimental results have shown that SWO is more efficient and effective (under time limits) than complete scheduling algorithms. We expect that this behaviour will persist even for larger problems as the one described in this paper.
- An e*vent ontology* would greatly facilitate entering the details of an event, since they could be retrieved from the predefined values of the event's class.
- *Overconstrained problems*. The system should always present a schedule to the user, even if it is not possible to satisfy all hard constraints. Suggestions to relax constraints should be provided.
- *Global scheduling*. The system should be able to change any past decision, in order to accommodate new events. For example, in order to schedule a new meeting, the system should be able to present options that include rescheduling existing events
- *Incremental scheduling*. Contrary to the previous requirement, the system should give the user the option to keep parts of its current schedule as fixed as possible. For this, criterion C_5 is relevant.
- *Adaptivity*. The system should provide the expert user with a mean to define new constraints and preferences, over the decision variables. This can be achieved through a constraint programming language.

Thus, our next step is to develop a rigorous model and scheduling algorithms for the problem formulation described in this paper and to evaluate the alternative schedules using a subset of the suggested evaluation model. This constitutes our future work.

Acknowledgements

Thanks to Pauline Berry and Bart Peintner for discussions. The work of the second author was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-07-D-0185/0004. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, or the Air Force Research Laboratory.

References

Bellotti, V., Dalal, B., Good, N., Flynn, P., Bobrow, D.G., and Ducheneaut, N. 2004. What a to-do: Studies of task management towards the design of a personal task list manager. In *Proc. of CHI'04*, Vienna, Austria, pp. 735-742.

Berry, P., Conley, K., Gervasio, M., Peintner, B., Uribe, T., and Yorke-Smith, N. 2006. Deploying a personalized time management agent. In *Proc. of AAMAS'06 Industrial Track*, Hakodate, Japan, pp. 1564-1571.

Berry, P., Donneau-Golencer, T., Duong, K., Gervasio, M., Peintner, B., and Yorke-Smith. N. 2008. Emma: An event management assistant. In *ICAPS'08 System Demos*, Sydney, Australia.

Berry, P., Donneau-Golencer, T., Duong, K., Gervasio, M., Peintner, B., and Yorke-Smith, N. 2009. Evaluating user-adaptive systems: Lessons from experiences with a personalized meeting scheduling assistant. *Proc. of IAAI'09*, Pasadena, CA.

Berry, P., Gervasio, M., Peintner, B., and Yorke-Smith, N. 2007. *Balancing the Needs of Personalization and Reasoning in a User-Centric Scheduling Assistant*. Technical Note 561, Artificial Intelligence Center, SRI International.

Conley, K. and Carpenter, J. 2007. Towel: Towards an intelligent to-do list. In *Proc. of AAAI Spring Symposium on Interaction Challenges for Artificial Assistants*, Stanford, CA, pp. 227-236.

Hiatt, L., Zimmerman, T., Smith, S.F., and Simmons, R. 2009. Strengthening schedules through uncertainty analysis. In *Proc. of IJCAI'09*, Pasadena, CA.

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

Keeney, R.L. and Raiffa, H. 1976. *Decision with Multiple Objectives*. John Wiley, New York.

Labreuche, C. and Grabisch, M. 2003. The Choquet integral for the aggregation of interval scales in multicriteria decision making. *Fuzzy Sets and Systems*, 137(1).

Maheswaran, R.T., Pearce, J.P., Varakantham, P., Bowring, E., and Tambe M. 2005. Valuations of Possible States (VPS): a quantitative framework for analysis of privacy loss among collaborative personal assistant agents. In *Proc. of AAMAS'05*, Utrecht, The Netherlands, pp. 1030-1037.

Mitchell, T.M., Caruana, R., Freitag, D., McDermott, J., and Zabowski, D. 1994. Experience with a learning personal assistance. *Comm. of the ACM*, 37(7):80-91.

Modi, P.J., Veloso, M., Smith, S.F., and Oh, J. 2004. CMRadar: A personal assistant agent for calendar management. In *Proc. of 6th Intl. Workshop on Agent-Oriented Information Systems* (AOIS'04), Riga, Latvia, pp. 134-148.

Refanidis, I. 2007. Managing personal tasks with time constraints and preferences. *Proc. of ICAPS'07*, Providence, RI, pp. 272-279.

Refanidis, I. and Alexiadis, A. 2008. SELFPLANNER: Planning your time! In *Proc. of ICAPS'08 Workshop on Scheduling and Planning Applications*, Sydney, Australia.

Sen, S. and Durfee, E.H. 1998. A formal study of distributed meeting scheduling. *Group Decision and Negotiation*, 7:265-289.

Weld. D. 1994. An Introduction to Least Commitment Planning. *AI Magazine*, 15(4): 27-61.