

A Constraint-Based Approach to Scheduling an Individual's Activities

IOANNIS REFANIDIS

University of Macedonia

and

NEIL YORKE-SMITH

SRI International, and American University of Beirut

The goal of helping to automate the management of an individual's time is ambitious in terms both of knowledge engineering and of the quality of the plans produced by an AI system. Modeling an individual's activities is itself a challenge, due to the variety of activity, constraint, and preference types involved. Activities might be simple or interruptible; they might have fixed or variable durations, constraints over their temporal domains, and binary constraints between them. Activities might require the individual being at specific locations in order, whereas traveling time should be taken into account. Some activities might require exclusivity, whereas others can be overlapped with compatible concurrent activities. Finally, while scheduled activities generate utility for the individual, extra utility might result from the way activities are scheduled in time, individually and in conjunction.

This article presents a rigorous, expressive model to represent an individual's activities, that is, activities whose scheduling is not contingent on any other person. Joint activities such as meetings are outside our remit; it is expected that these are arranged manually or through negotiation mechanisms and they are considered as fixed busy times in the individual's calendar. The model, formulated as a constraint optimization problem, is general enough to accommodate a variety of situations. We present a scheduler that operates on this rich model, based on the general squeaky wheel optimization framework and enhanced with domain-dependent heuristics and forward checking. Our empirical evaluation demonstrates both the efficiency and the effectiveness of the selected approach. Part of the work described has been implemented in the SELFPLANNER system, a Web-based intelligent calendar application that utilizes Google Calendar.

Categories and Subject Descriptors: I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*scheduling; plan execution, formation, generation; heuristic methods*

General Terms: Algorithms

The work of N. Yorke-Smith was supported in part by the Defence Advanced Research Projects Agency (DARPA) under Contract No. FA8750-07-D-0185/0004 at SRI International. 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.

Authors' addresses: I. Refanidis, University of Macedonia, Greece; email: yrefanid@uom.gr; N. Yorke-Smith, American University of Beirut, Lebanon; email: nysmith@aub.edu.lb.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2010 ACM 2157-6904/11-ART12 \$10.00
DOI 10.1145/1869397.1869401 <http://doi.acm.org/10.1145/1869397.1869401>

Additional Key Words and Phrases: Greedy algorithms, constraints, intelligent calendar applications, activity modeling, preferences

ACM Reference Format:

Refanidis, I. and Yorke-Smith, N. 2010. A constraint-based approach to scheduling an individual's activities. *ACM Trans. Intell. Syst. Technol.* 1, 2, Article 12 (November 2010), 32 pages.

DOI = 10.1145/1869397.1869401 <http://doi.acm.org/10.1145/1869397.1869401>

1. INTRODUCTION

Developing an intelligent tool to assist an individual in the management of his or her personal time is an ambitious goal for at least three reasons. First, ethnographic studies show that people seek appropriate, contextualized assistance [Palen 1999]. Second, much knowledge is implicit and not available to an AI (Artificial Intelligence) reasoning system, or is informally represented. Third, the reasoning task of producing suggested plans must account for complex and subtle preferences and constraints.

While an ambitious end goal, intelligent assistance with time and task management is a recognized target for AI [Refanidis et al. 2004; Myers et al. 2007; Freed et al. 2008]. Although electronic calendaring tools, such as Microsoft Outlook and Google Calendar, offer multiple forms of time management assistance, their provision of automatic or semi-automatic scheduling of users' activities is limited. More common functionality is assistance in adding events into the user's and others' calendars, detection of conflicts, task assignment, meeting arrangements, and calendar sharing and publishing.

Our contribution in this article is a modest step towards the end goal. We investigate how constraint-based reasoning can generate a plan for a set of activities for an individual, taking into account interruptible activities, variable durations, alternative locations and traveling times, as well as a variety of intra- and interactivity constraints and user's preferences. We address the representation challenges with an expressive and rigorous constraint-based model. In this article we do not treat the knowledge acquisition challenges.

This article address the reasoning challenges with an adaptation of the incomplete heuristic search framework *Squeaky Wheel Optimization* (SWO) [Joslin and Clements 1999], enhanced with domain-dependent heuristics and forward checking to make more informed choices. Computation time for solving the problem is limited; this is the reason to adopt an incomplete but fast solving algorithm. A further reason is that an envisioned system may employ a server-based model accessed simultaneously by many users using low-cost devices, and thus the server's overall workload is a consideration. The presented approach was tested experimentally through a large set of benchmark problem instances, which showed it to be remarkably efficient and effective, producing plans that are near optimal. We suppose that activities to be scheduled are given; we do not consider scheduling with resources; and we explicitly exclude from our scope interperson coordination and meeting arrangement.

The work described in this article is being implemented in `SELFPLANNER`,¹ a Web-based intelligent calendar application that utilizes Google Calendar. An earlier version of the system, implementing the model described in Refanidis [2007], has been described in Refanidis and Alexiadis [2008]. The present article does not address the functionalities, usability considerations, and adoption of a deployed tool. We understand that the definition of a scheduling problem with such an expressive model requires significant effort from the individual. Aspects of this problem have been addressed in Alexiadis and Refanidis [2009]. However, in addition to usability and knowledge acquisition issues, there are several open engineering issues that need to be addressed, which constitute part of our future work.

This article significantly extends the work described in Refanidis et al. [2006] and Refanidis [2007], by presenting an enhanced activity model and extending the algorithms proposed there in order to cope with the new features. An even more general model was presented by Refanidis and Yorke-Smith [2009] in a more informal setting. The present article gives a rigorous definition for a subset of the features presented there and develops effective and efficient solution techniques.

The rest of the article is organized as follows. Section 2 introduces the problem formulation and Section 3 describes our solving algorithm. Section 4 presents experimental results on fresh benchmark problem instances. Section 5 situates our work in the literature. Finally, Section 6 concludes and sets direction for future work.

2. PROBLEM FORMULATION

This section formulates the underlying computational problem. First we present the motivations behind our choices, that is, specific cases of activities that the proposed model intends to model, and then we give the rigorous formulation.

2.1 Motivation

There are several types of an individual's activities. The simplest types concern a totally inflexible activity, as for example attending a lecture as a student. This activity has a prespecified time and location reference, as well as a fixed duration. The individual only needs to decide whether to attend the lecture or not. However, not all activities are so simple like this one.

Some activities do not prespecify their start time, as for example going to a theater performance. Although the duration and the location of this activity are known in advance, the individual has the option to decide the day (and perhaps the time) to attend. A deadline might also exist in this case, for example, the last scheduled performance. Another example of such a *fluent* activity is having lunch at a restaurant. This activity is characterized by a specific location and duration, but the individual decides the exact time frame to have lunch, within the working hours of the restaurant. A temporal domain, consisting

¹<http://selfplanner.uom.gr>

of disjoint temporal intervals, accompanies every fluent activity. Furthermore, the duration of an activity is not always fixed. In some cases, as in the lunch example, the individual might decide on the duration, with more duration resulting in more satisfaction. Bounds on the minimum and maximum duration might exist; lunch cannot extend all day, even if the restaurant stays open!

An activity may be interruptible. Examples of interruptible activities are reading, doing homework, or writing a paper. Bounds on the durations of the parts of an interruptible activity might exist as well. For example, suppose that there is no point in scheduling a part of less than one hour or more than eight hours of an activity concerning writing a paper; the author will be inefficient in both cases. Furthermore, the individual might want a minimum temporal distance between parts of the same activity, for example, in order to relax. Or, she might want a maximum temporal distance between parts of the same activity. For example, in the case of writing a paper, the author might want to finish the work within two months from its beginning, irrelevant to when this activity actually began.

Activities might have alternative locations where they can be performed. For example, for a “weekly shopping” activity, the individual might have the option to select between a set of alternative malls, with the best choice depending on other activities scheduled adjacent in time. Reasoning about alternative locations requires considering transportation, with the obvious goal to minimize the time devoted to it. Interruptible activities with multiple locations do not require all of their parts to be scheduled at the same location. For example, for an activity concerning preparation of a presentation, some of the work can be scheduled at office and the rest at home.

Most activities require the full attention of the individual. However, there are activities that do not require such exclusivity; they can be scheduled in parallel with others, provided that the individual is capable of performing them simultaneously and a location compatible with both activities exists. Examples of situations with overlapping activities are attending a teleconference while reading/writing emails, or escorting the children to some class and waiting for them while reading a book. In an individual’s calendar we may also find dummy or informative activities, which are activities with zero utilization. For example, one could schedule a dummy activity concerning attending ICAPS 2011 conference in Freiburg, thus allowing in the plan only activities that can take place in Freiburg to be scheduled in the same time period.

We consider three types of binary constraints that might exist between pairs of activities. First, *ordering constraints* specify one activity must occur before another, such as, buying tickets before going to the theater performance. Next, *proximity constraints* impose minimum and/or maximum bounds on the temporal distance between different activities. For example, one might avoid scheduling intensive activities such as reading or writing a paper close to teaching or other heavy activities. Finally, *implication constraints* might exist between activities, usually denoting prerequisites (not necessarily temporal). For example, if you borrow a book from the library, you must return it.

Scheduling an individual’s activities is more than a constraint satisfaction problem: it is an optimization problem. In many cases not all of an individual’s

- Activities
 - Duration
 - Interruptibility
 - Intra-activity proximity constraint
 - Constraint parameters
 - Temporal domain
 - Compatible locations
 - Utilization
- Binary constraints (ordering, proximity, implication)
 - Involved activities
 - Constraint parameters
- Utility sources
 - Activity's utility
 - Duration utility profile
 - Intra-activity proximity preference
 - Constraint Parameters
 - Utility profile
 - Temporal utility profile
 - Binary preferences (ordering, proximity, implication)
 - Involved activities
 - Constraint parameters
 - Utility profile

Fig. 1. An outline of the attributes, constraints, and preferences of the proposed model.

activities can fit compatibly into her calendar; hence a selection must be made among them, while satisfying all involved constraints. Since all activities are not of equal importance, we assign a utility to each one. This utility might be fixed, or dependent on the duration of the activity and the way it has been scheduled within its temporal domain. Furthermore, extra utility may arise from the way the activities have been scheduled in relation to each other. Ultimately, it is up to the user to assign utilities to the various constraints, thus turning the constraints into preferences that might be satisfied only to a degree by the resulting plan. Depending on the actual utility values, we expect to obtain plans with specific activities left out (although it would be possible to fit them in the plan) in order to have a better overall plan for the scheduled activities. Assigning utilities to the various aspects of the activities is in itself an involved problem; it is not within the scope of this article.

Figure 1 outlines the main entities of the proposed model: activities, binary constraints, and the various utility sources. The first two define the constraint satisfaction problem and the last defines the constraint optimization problem. The details are given next in Section 2.2.

2.2 Formulation

Time management tools consider time to be discrete. Indeed, most electronic organizers use a minimum time slot, usually 15 or 30 minutes. Hence, we model time as a nonnegative integer, with zero denoting the current time. The next

subsections present in detail the proposed model for an individual's activities, with the order of presentation following the outline of Figure 1.

2.2.1 Activity Modeling. Consider a set T of N activities, $T = \{T_1, T_2, \dots, T_N\}$, each one being characterized by a set of properties. First, we suppose that activity T_i has a duration range

$$[dur_i^{\min}, dur_i^{\max}] \quad (1)$$

with meaning that there is no value in scheduling T_i in the plan with an allocated duration less than dur_i^{\min} (i.e., it would be preferable not to schedule T_i at all) and no extra value in scheduling T_i in the plan with an allocated duration more than dur_i^{\max} . Writing dur_i to denote the duration actually allocated to T_i , we have

$$\forall T_i, \quad dur_i^{\min} \leq dur_i \leq dur_i^{\max} \quad \text{or} \quad dur_i = 0, \quad (C1)$$

where (Ci) is interpreted as ‘‘Constraint i’’. An activity T_i may be (but need not be) *interruptible*, that is, it may be possible to split the activity into parts that can be scheduled separately. With the decision variable p_i we denote the number of parts, $p_i \geq 1$, into which T_i has been split in the plan (for noninterruptible activities we always have $p_i = 1$). With T_{ij} we denote the j^{th} part of the i^{th} activity, $1 \leq j \leq p_i$. For each part T_{ij} we introduce the decision variables t_{ij} and dur_{ij} , denoting the start time and the duration of part j , respectively. The sum of the durations of all parts of an activity equals dur_i :

$$\forall T_{ij}, \quad \sum_{j=1}^{p_i} dur_{ij} = dur_i \quad (C2)$$

For each interruptible activity T_i , the minimum and maximum allowed duration for each of its parts, smi_n_i and $smax_i$, as well as the minimum and maximum allowed temporal distance between each pair of its parts, dmi_n_i and $dmax_i$, are given. In particular, for the durations of the parts of an interruptible activity we have

$$\forall T_{ij}, \quad smi_n_i \leq dur_{ij} \leq smax_i. \quad (C3)$$

Depending on the values of smi_n_i and $smax_i$ and the overall duration range of the activity (Eq. (1)), implicit constraints are inferred on the values of p_i for interruptible activities. For example, if $smi_n_i \geq dur_i^{\max}/2$, then $p_i \leq 2$. Note that for noninterruptible activities, it is reasonable to expect that $smi_n_i = dur_i^{\min}$ and $smax_i = dur_i^{\max}$, whereas for interruptible activities it is expected that $smi_n_i \leq dur_i^{\min}$ and $smax_i \leq dur_i^{\max}$.

Regarding the constraints on the temporal distance between the parts of an interruptible activity T_i , we have minimum and maximum distance constraints

$$\forall T_{ij}, T_{ik}, j \neq k \Rightarrow t_{ij} + dur_{ij} + dmi_n_i \leq t_{ik} \vee t_{ik} + dur_{ik} + dmi_n_i \leq t_{ij} \quad (C4)$$

and

$$\forall T_{ij}, T_{ik}, j \neq k \Rightarrow t_{ij} + dmax_i \leq t_{ik} + dur_{ik} \wedge t_{ik} + dmax_i \leq t_{ij} + dur_{ij} \quad (C5)$$

with the symbols \vee and \wedge denoting disjunction and conjunction, respectively. The special case with $dmin_i = 0$ and $dmax_i = \infty$ denotes the absence of any proximity constraint between the parts of the interruptible activity T_i . Note that proximity constraints should hold for any pair of infinitesimal pieces of the interruptible activity; thus, the closest ends between any two parts of the interruptible activity are considered for the minimum distance constraint (C4), whereas the remotest ends are considered for the maximum distance constraint (C5).

Each activity T_i has a temporal domain $D_i = [a_{i_1}, b_{i_1}] \cup [a_{i_2}, b_{i_2}] \cup \dots \cup [a_{i_{F_i}}, b_{i_{F_i}}]$, consisting of a set of intervals, with F_i denoting their number. The temporal domain determines the time periods when the parts of an activity can be scheduled.

$$\forall T_{ij}, \exists k, 1 \leq k \leq F_i : a_{ik} \leq t_{ij} \leq b_{ik} - dur_{ij} \quad (C6)$$

It is expected that $a_{ij} + smin_i \leq b_{ij}$ as well as $b_{ij} < a_{i,j+1}$ for each $1 \leq i \leq N, 1 \leq j \leq F_i$.

A set of M locations, $Loc = \{L_1, L_2, \dots, L_M\}$ is given, and a two-dimensional matrix $Dist$ of the temporal distances between locations (nonnegative integers), not necessarily symmetric. Each activity T_i is associated with a nonempty set of locations, $Loc_i \subseteq Loc$, denoting alternative places where the individual should be in order to perform each part of the activity. The decision variable $l_{ij} \in Loc_i$ denotes the particular location where part T_{ij} is scheduled. For parts of the same or different activities scheduled at distant locations, the following condition should hold.

$$l_{ij} \in Loc_i \quad (C7)$$

$$\begin{aligned} &\forall T_{ij}, T_{mn}, T_{ij} \neq T_{mn} \wedge (Dist(l_{ij}, l_{mn}) > 0 \vee Dist(l_{mn}, l_{ij}) > 0) \\ &\Rightarrow t_{ij} + dur_{ij} + Dist(l_{ij}, l_{mn}) \leq t_{mn} \vee t_{mn} + dur_{mn} + Dist(l_{mn}, l_{ij}) \leq t_{ij} \end{aligned} \quad (C8)$$

Note that (C8) imposes disjointness only for parts of activities scheduled at distant locations. Parts of activities scheduled at nondistant locations may overlap, depending on their utilization values. A special location with no distance from any other location, called *anywhere*, can be defined, that is, $Dist(anywhere, L) = Dist(L, anywhere) = 0$ for any $L \in Loc$ (of course, *anywhere* is also a member of Loc). Assigning *anywhere* to an activity means that the activity can be performed at any location, so there is no reason to assign any other location to the same activity.

Each activity T_i is characterized by a utilization value, denoted $utilization_i$, such that $0 \leq utilization_i \leq 1$. Utilization expresses the percentage of an individual's attention an activity requires while it is performed. Two or more activities may overlap in time provided they are scheduled at nondistant locations and the sum of their utilization values does not exceed 100%.

$$\forall t, \sum_{\substack{T_{ij} \\ t_{ij} \leq t < t_{ij} + dur_{ij}}} utilization_i \leq 1 \quad (C9)$$

The proposed model does not allow parts of the same activity to overlap, even if the sum of their utilizations does not exceed one (constraint C4, even for $dmin_i = 0$).

2.2.2 Binary Constraints. The proposed model allows for *binary constraints* between activities. In particular, it supports *ordering* constraints, *minimum distance* and *maximum distance proximity* constraints, and *implication* constraints. For convenience, ordering and proximity constraints are also referred within this article as *temporal constraints*. All these constraints are hard, that is, any plan violating any binary constraint is considered invalid. Binary temporal constraints are conditional on the presence of both activities in the plan; in case any of the involved activities is not scheduled, the temporal constraint is not applicable.

An ordering constraint between activities T_i and T_j , denoted $T_i < T_j$, is interpreted as

$$\forall T_i, T_j, T_i < T_j \Leftrightarrow dur_i > 0 \wedge dur_j > 0 \Rightarrow \forall T_{ik}, T_{jl}, t_{ik} + dur_{ik} \leq t_{jl}. \quad (C10)$$

In other words, no part of T_j can be scheduled before all parts of T_i have finished.

Concerning minimum distance and maximum distance constraints, let $dmin_{ij} = dmin_{ji}$ and $dmax_{ij} = dmax_{ji}$ nonnegative integers denoting the minimum and maximum allowed distances between any infinitesimal pieces of the two activities. For any pair of parts, T_{ik} of T_i and T_{jl} of T_j , among which minimum and maximum distance constraints exist, the following two conditions should hold respectively.

$$\forall T_{ik}, T_{jl}, t_{ik} + dur_{ik} + dmin_{ij} \leq t_{jl} \vee t_{jl} + dur_{jl} + dmin_{ij} \leq t_{ik} \quad (C11)$$

$$\forall T_{ik}, T_{jl}, t_{ik} + dmax_{ij} \leq t_{jl} + dur_{jl} \wedge t_{jl} + dmax_{ij} \leq t_{ik} + dur_{ik} \quad (C12)$$

The special cases where $dmin_{ij} = 0$ or $dmax_{ij} = \infty$ denote the absence of the corresponding proximity constraint between activities T_i and T_j .

Implication constraints between pairs of activities T_i and T_j , denoted with $T_i \Rightarrow T_j$, are interpreted as follows: T_j should be scheduled, in order for T_i to be scheduled as well, that is, T_j is a prerequisite for T_i . Note however that T_j may be scheduled and T_i not.

$$\forall T_i, T_j, T_i \Rightarrow T_j \Leftrightarrow dur_i > 0 \Rightarrow dur_j > 0 \quad (C13)$$

An implication constraint does not by itself impose any ordering constraint between the two activities, that is, T_j might be scheduled before or after T_i .

2.2.3 Utility Sources. Scheduling an individual's activities is considered an optimization problem in the proposed model. By this we mean that the goal is not to schedule all the activities, but to maximize the utility of the plan. Accordingly, an empty plan is a valid plan, but usually has zero utility. There are several aspects of the problem that contribute to the utility; we distinguish three of them: which activities are included in the plan, how each activity is scheduled individually, and how the activities are scheduled in relation to each other. All utilities are considered nonnegative real numbers.

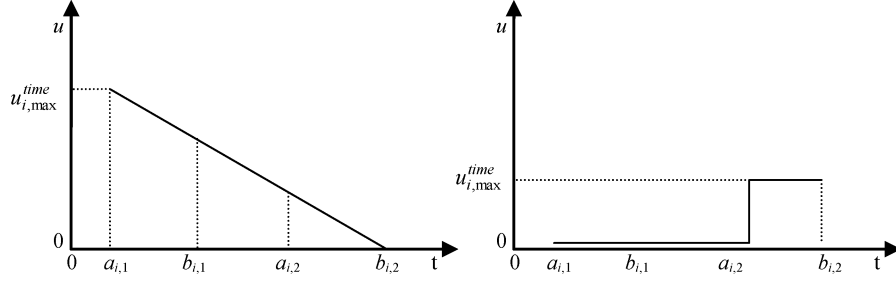


Fig. 2. (a) A linear descending domain utility function (the earliest the better). (b) A stepwise ascending domain utility function (better if later than a specific time point).

In the following paragraphs we exemplify the proposed model by defining utility functions for the various utility sources. Our choices are motivated by simplicity and efficiency. However, simple modifications will extend the algorithmic solution we propose in Section 3 to more complicated utility functions.

It is expected that each activity being included in the plan yields an amount of utility. This utility may depend on the allocated duration, while satisfying (C1). Assuming that the received utility does not decrease with the duration, we adopt a piecewise linear model for T_i 's utility, $U_i(dur_i)$, defined as follows.

$$U_i(dur_i) = \begin{cases} 0, & \text{if } dur_i < dur_i^{\min} \\ u_i^{low} + (dur_i - dur_i^{\min}) / (dur_i^{\max} - dur_i^{\min}) \cdot (u_i^{high} - u_i^{low}), & \text{if } dur_i^{\min} \leq dur_i \leq dur_i^{\max} \\ u_i^{high}, & \text{if } dur_i > dur_i^{\max} \end{cases} \quad (2)$$

where u_i^{low} and u_i^{high} are utility values, with $0 \leq u_i^{low} \leq u_i^{high}$.

Preferences over an activity's temporal domain are allowed. For each activity T_i we define a function u_i^{time} , which for any time point t returns the utility received by scheduling a unit of T_i 's duration at that point. In the proposed model we allow for three types of temporal domain utility functions: Constant utility, linear utility, and stepwise utility. For linear (stepwise) domain utility functions we distinguish ascending functions, that is, the latest the better (better after the step time point) and descending functions, that is, the earliest the better (better before the step time point). Figure 2(a) shows a linear descending function whereas Figure 2(b) shows a stepwise ascending function. Taking into account that T_i might be interruptible, as well as that T_i 's duration might vary, we define the accumulated utility received by the way T_i is scheduled in time with the following integral

$$U_i^{time}(\pi_i) = \frac{\sum_{j=1}^{p_i} \int_{t_{ij}}^{t_{ij} + dur_{ij}} u_i^{time}(t) dt}{\sum_{j=1}^{p_i} dur_{ij}}, \quad (3)$$

where π_i is the current plan of T_i . The integral in (3) is over the various time intervals $[t_{ij}, t_{ij} + dur_{ij}]$ where some part of T_i has been scheduled (there are p_i such parts). Hence, Eq. (3) defines the utility received by the way an activity is scheduled within its domain as the average preference of the time slots allocated to the activity.

In the case of interruptible activities, the proposed model allows for intraactivity proximity preferences. Assuming a minimum and a maximum desired distance for the parts of an interruptible activity, denoted with $pdmin_i$ and $pdmax_i$ respectively, as well as the corresponding maximum utilities u_i^{dmin} and u_i^{dmax} received in case of full satisfaction, we define the utility received by the user with respect to intraactivity proximity preferences as follows.

$$U_i^{dmin}(\pi_i) = \frac{\iint_{t \in \pi_i, t' \in \pi_i, \text{abs}(t-t') \geq pdmin_i} dt dt'}{dur_i \times dur_i} \times u_i^{dmin} \quad (4)$$

$$U_i^{dmax}(\pi_i) = \frac{\iint_{t \in \pi_i, t' \in \pi_i, \text{abs}(t-t') \leq pdmax_i} dt dt'}{dur_i \times dur_i} \times u_i^{dmax} \quad (5)$$

Formulas (4) and (5) take into account partial satisfaction of the underlying constraint. Indeed, the utility received from the proximity preferences is equal to the fraction of the pairs of infinitesimal pieces of T_i for which the underlying constraint holds, to the total number of pairs of infinitesimal pieces, times the maximum utility assigned to each proximity preference. It is worth noting that proximity constraints do not preclude proximity preferences, provided that $pdmin_i > dmin_i$ and $pdmax_i < dmax_i$.

Example 1. Suppose an interruptible activity of reading a book, with a fixed duration of 24 hours, scheduled in three parts of 8 hours. Suppose that there is a maximum distance preference of reading the book within a week; however, only two of the activity's three parts have been scheduled within a week, with the third part being scheduled one month later. Finally, suppose that $u_i^{dmax} = 1$, that is, we get one unit of utility in case of full satisfaction of the preference.

To simplify our calculations, we discretize the integral of Eq. (5) using 30 minutes as the unit of time. Hence there are 32 infinitesimal pieces of the activity scheduled within one week, corresponding to the first two parts, as well as additional 16 pieces, corresponding to the third part. The pairs of infinitesimal pieces scheduled within one week distance are 32×32 plus 16×16 , that is, 1280 pairs. The duration of the activity is 48 units of time, so Eq. (5) evaluates to about 0.556 units of utility.

Utility may also result from the way activities are scheduled in relation to each other. The proposed model supports four types of binary preferences: ordering preferences, minimum distance preferences, maximum distance preferences, and implication preferences. For convenience, the first three of them are also referred as temporal preferences within this article, whereas the distance preferences are also referred as proximity preferences.

Concerning temporal preferences, the proposed model allows for partial satisfaction. To this end we define the notion of *Degree of Satisfaction* of preference p , as the fraction of the pairs of infinitesimal pieces of the two activities for which the preference holds to the total number of pairs of infinitesimal pieces

of the two activities.

$$DoS(p) = \frac{\iint_{t_i \in \pi_i, t_j \in \pi_j, p(t_i - t_j) \text{ holds}} dt_i dt_j}{dur_i \times dur_j} \quad (6)$$

DoS is a continuous integral taking values between 0 and 1; however, it can be approximated by a discrete integral with dt_i and dt_j being equal to 1. Having computed $DoS(p)$ for a binary preference p , the utility received by this preference is defined as

$$U_p = u_p \times DoS(p), \quad (7)$$

where u_p the maximum possible utility that results in case of full satisfaction of p .

We use the notation $u_{i < j}$ to refer to the utility received in case T_i has been entirely scheduled before T_j ; $pdmin_{ij}$ and $pdmax_{ij}$ to refer to the preferred minimum and maximum distances between activities T_i and T_j , and u_{ij}^{dmin} and u_{ij}^{dmax} to refer to the resulting utilities in case of full satisfaction, respectively. Note finally that temporal preferences do not yield any utility in case any one of the involved activities is not included in the plan, so DoS is equal to 0 in these cases.

Example 2. One is attending two seminars, A and B, with having attended A being useful to better understand B, so an ordering preference between the two activities exists. Each seminar consists of two 120 minute lectures, say A_1, A_2, B_1 and B_2 . Suppose that the ordering of the lectures is $A_1 < B_1 < A_2 < B_2$. Assuming again a time slot of 30 minutes, the degree of satisfaction for this ordering preference is computed as follows: The 4 units of A_1 satisfy the preference regarding the 8 units of B. The 4 units of A_2 satisfy the preference regarding only the 4 units of B_2 . In total, there are 4×8 plus 4×4 , that is, 48 pairs of unit size infinitesimal pieces of A and B that satisfy the preference. The total number of pairs of unit size infinitesimal pieces of A and B is 4×8 , that is, 64. Hence, the degree of satisfaction evaluates to 0.75.

The way we defined the degree of satisfaction is not the only possibility. A more general way would be to allow arbitrary powers x of the current definition, thus penalizing stronger the partial satisfaction with higher values of x . In the extreme case where $x \rightarrow \infty$, partial satisfaction would not yield any utility. It is in our future plans to extend the proposed model and the solution framework towards this direction.

Implication preferences of the form $T_i \Rightarrow T_j$ are interpreted as a preference to include T_j in the plan every time T_i is included as well. Since an activity is either included or not included in the plan, implication preferences do not allow for partial satisfaction. Thus any utility $u_{T_i \Rightarrow T_j}$ assigned to the implication preference is accumulated to the overall utility in all cases except the case where T_i is in the plan whereas T_j is not, that is, the only case an implication is violated. Note that implication preferences may yield utility even for an empty plan.

Finally, it is interesting to note that, with the exception of proximity constraints, binary constraints preclude the corresponding binary preferences. For

example, there is no point in having a preference of the form $T_i < T_j$ or $T_i \Rightarrow T_j$, in the presence of a constraint of the same type between the same activities. However, in case of proximity constraints, it is reasonable to have both a proximity constraint and a proximity preference of the same type between the same activities, provided that the distance imposed by the preference is tighter than the distance imposed by the constraint, that is, $pdmin_{ij} > dmin_{ij}$ or $pdmax_{ij} < dmax_{ij}$.

To summarize, the optimization problem is formulated as follows.

Given:

- a set of N activities, $T = T_1, T_2, \dots, T_N$, each one of them characterized by its duration range, duration utility profile, temporal domain, temporal domain preference function, utilization, a set of alternative locations, interruptibility property, minimum and maximum part sizes, as well as required minimum and maximum part distances for interruptible activities, preferred minimum and maximum part distances, and the corresponding utilities;
- a two-dimensional matrix with temporal distances between all locations;
- a set C of binary constraints (ordering, proximity, and implication) over the activities; and
- a set P of binary preferences (ordering, proximity, and implication) over the activities.

Schedule:

the activities in time and space, by deciding the values of their start times t_{ij} , their durations dur_{ij} , and their locations l_{ij} , while trying to maximize the following quantity

$$U = \sum_{\substack{i \\ dur_i \geq dur_i^{\min}}} (U_i(dur_i) + U_i^{time}(\pi_i) + U_i^{dmin}(\pi_i) + U_i^{dmax}(\pi_i)) + \sum_{p(T_i, T_j) \in P} u_p \times DoS(p(T_i, T_j)) \quad (8)$$

subject to constraints (C1) to (C13).

3. METAHEURISTIC GREEDY SOLUTION

Having defined our model of the problem of scheduling an individual's activities, in this section we present a greedy approach to tackle the problem, while trying to maximize the received utility. The proposed approach is based on the squeaky wheel optimization framework. We start with a brief overview of this framework and then explain its adaptation to the problem at hand. Figure 4 summarizes this section by presenting the overall approach.

3.1 The Squeaky Wheel Optimization Framework

Squeaky Wheel Optimization (SWO) is a general optimization framework that can be adapted to several constraint satisfaction and optimization problems

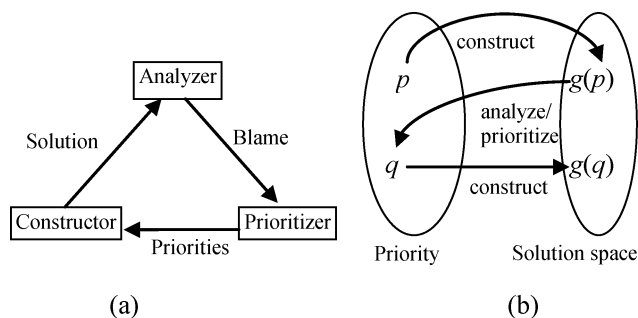


Fig. 3. (a) The construct/analyze/prioritize cycle. (b) Coupled search spaces.

[Joslin and Clements 1999]. The core of SWO is a construct/analyze/prioritize cycle, as shown in Figure 3(a). Constraint variables are ordered in a priority queue in decreasing order of an initial estimate of the difficulty to assign a value to each one of them. A solution is constructed by a greedy algorithm, taking decisions in the order determined by the priority queue. This solution is then analyzed to find those constraint variables that were the “trouble makers”. The priorities of the “trouble makers” are increased, causing the greedy constructor to deal with them sooner in the next iteration. This cycle repeats until some termination condition occurs.

SWO is known to be a fast but incomplete search procedure. As shown in Figure 3(b), SWO searches in two coupled spaces: the priority space and the solution space. The greedy construction algorithm defines a function g from the priority queues to the solutions, that is, for each ordering p of the activities a plan $g(p)$ is defined. However, function g may be neither surjective nor injective; so, many feasible solutions may not correspond to any ordering of the activities in the queue.

Managing an individual’s activities is considered an optimization problem in this article. Hence, SWO should focus more on the utility that is estimated to result from the various scheduling options rather than the difficulty to schedule each activity, although the latter is also relevant. The next subsections give the details of the proposed SWO adaptation to tackle the problem formulated in Section 2.

3.2 Assessing the Difficulty to Schedule an Activity

Crucial to all phases of SWO is an efficient metric to estimate the difficulty to schedule an activity. Taking into account the noncontinuous temporal domains, as well as the constraints on the sizes and the distances of the parts on an interruptible activity, we designed metric m_3 (two simpler metrics called m_1 and m_2 were presented in Refanidis et al. [2006]). The new metric attempts to estimate how much duration can be scheduled with an activity’s temporal domain, taking into account the extra constraints for interruptible activities. A precise answer to whether an activity fits within its temporal domain may be a hard problem. Our goal is an upper bound of the maximum possible duration

1. Let $P^* = \emptyset$ be the best plan found so far and let $U^* = 0$ be its utility.
 2. Initialize the priority queue with the activities placed in decreasing order of their initial utility estimate \hat{U}_i .
 3. **While** no termination condition is met,
 - a. **While** the priority queue is not empty,
 - i. Extract the next activity T_i from the priority queue. Ignore (i.e. extract without further processing) activities T_j for which there was an activity T_k earlier in the priority queue, such that an implication constraint $T_j \Rightarrow T_k$ exists and T_k was unable to schedule.
 - ii. Compute \hat{U}_j for each one of the subsequent activities T_j in the queue.
 - iii. **If** T_i is not interruptible
 1. Decide the variables $\langle t_i, l_i, dur_i \rangle$ so that \hat{U} is maximized. Employ forward checking for each alternative decision. Ignore decisions that make any subsequent activity T_j , for which an implication constraint $T_i \Rightarrow T_j$ exists, unschedulable.
 2. **If** T_i has been scheduled, apply forward checking for the taken decision.
 else
 1. Let $dur_i = 0$
 2. **While** $dur_i < dur_i^{\min}$,
 - a. Schedule the next part T_{ij} by deciding the variables $\langle t_{ij}, l_{ij}, dur_{ij} \rangle$ so that \hat{U} is maximized. Employ forward checking for each alternative decision. Ignore infeasible values for dur_{ij} . Ignore decisions that make any subsequent activity T_j , for which an implication constraint $T_i \Rightarrow T_j$ exists, unschedulable.
 - b. **If** T_{ij} cannot be scheduled, retract any previously scheduled part of T_i and go to step v.
 - c. Increase dur_i by dur_{ij} . Apply forward checking for the taken decision.
 3. **While** $dur_i < dur_i^{\max}$,
 - a. Schedule additional parts T_{ij} by deciding the variables $\langle t_{ij}, l_{ij}, dur_{ij} \rangle$ so that \hat{U} is increased wrt the case of not scheduling any additional part at all. Employ forward checking for each alternative decision. Ignore infeasible values for dur_{ij} . Ignore decisions that make any subsequent activity T_j , for which an implication constraint $T_i \Rightarrow T_j$ exists, unschedulable.
 - b. **If** $dur_{ij} > 0$
 - i. Increase dur_i by dur_{ij} . Apply forward checking for the taken decision.
 - iv. **If** T_i has been scheduled, compute \hat{U}'_i for each one of the remaining activities T_j . Mark any T_j , such that $\hat{U}'_j < \hat{U}_j$, for promotion before T_i .
 - v. **If** T_i has not been scheduled,
 1. Retract any consequence of forward checking while trying to schedule T_i .
 2. Retract recursively other already scheduled activities T_k such that an implication constraint of the form $T_k \Rightarrow T_i$ exists. Retract the consequences of forward checking from scheduling T_k .
 - b. Compute U for the current plan P . If $U > U^*$, update P^* with P and U^* with U .
 - c. Promote activities marked for promotion.
 - d. Reinitialize the temporal domains of the activities.
4. Return P^* and U^* .

Fig. 4. Overview of the metaheuristic greedy solution.

that can be scheduled within an activity's temporal domain; if this estimate is less than dur_i^{\min} for activity T_i , then T_i cannot be scheduled.

In simple words, m_3 works as follows. Starting from the left end a_{ik} of each interval $[a_{ik}, b_{ik}]$ of activity's T_i temporal domain D_i , schedule parts of the maximum possible duration d , such that $smi_n_i \leq d \leq smax_i$, leaving the minimum required distance between them $dmin_i$. Ignore the minimum distance constraint $dmin_i$ between parts that have been scheduled in different intervals of D_i , in order for the heuristic to remain admissible; otherwise, significant computational effort is required to ensure admissibility. Then use the ratio between the activity's minimum duration dur_i^{\min} to the scheduled duration as an estimate of the difficulty to schedule the activity. Clearly, if m_3 exceeds 1, the activity cannot be scheduled.

Example 3. Suppose an activity T_i with $dur_i^{\min} = 8$, $smi_n_i = 3$, $smax_i = 5$, $dmin_i = 3$ and $D_i = [0, 12] \cup [14, 20]$. Within the first interval we could schedule part T_{i1} from $t = 0$ to $t = 5$ ($d = 5$) and part T_{i2} from $t = 8$ to $t = 12$ ($d = 4$). Within the second interval we could schedule part T_{i3} from $t = 14$ to $t = 19$ ($d = 5$). Note that the minimum distance constraint among T_{i2} and T_{i3} is ignored. The totally scheduled duration is $\sum_j dur_{ij} = 14$, so $m_3 = 8/14 = 0.57$.

Metric m_3 could be considered a more elaborate version of metric m_1 presented in Refanidis et al. [2006] and defined as the ratio between the duration of an activity and the net size of its domain. Hence, m_3 dominates m_1 , since the net size of a domain is never smaller than the maximum possible duration that can be scheduled within a domain, as it is computed by m_3 . In Example 3 that came before, m_1 would evaluate to 0.4. On the other hand, m_3 does not dominate metric m_2 , also defined in Refanidis et al. [2006] as the ratio between the minimum possible makespan of the activity and the width of its domain. For large values of $dmin_i$ (compared to the sizes of the domain intervals and the gaps among them), it is expected that m_2 gives better estimates than m_3 , whereas for small values of $dmin_i$ it is expected the opposite. As an example of evaluating m_2 , in the particular case T_i requires at least two parts, for example, T_{i1} with $dur_{i1} = 5$ and T_{i2} with $dur_{i2} = 3$, so the minimum possible makespan of T_i (considering also the minimum distance $dmin_i = 3$ between T_{i1} and T_{i2}) is 11. The width of D_i is 20, so m_2 evaluates to $11/20 = 0.55$. Neither m_2 nor m_3 ever overestimate the maximum possible duration that can fit within an activity's temporal domain. Hence, taking the maximum value between m_2 and m_3 is the best strategy. So, we define the difficulty to schedule activity T_i as

$$diff(T_i) = \max(m_2(T_i), m_3(T_i)). \quad (9)$$

3.3 Estimating the Overall Utility

Since we are interested in the utility generated by a plan, it is crucial to be able to obtain estimates for it before grounding all the decision variables. There are several sources of utility, most of them requiring the activity to be schedulable (implication preferences are the only exception). Hence, estimating the utility

that will result by scheduling a set of activities is related to assessing the difficulties to schedule them.

We distinguish two sources of utility: unary and binary preferences. Further, there are three states for an activity under evaluation: the activity has already been decided (i.e., it has been scheduled or it has been decided not to be included in the plan), the (interruptible) activity is partially scheduled, and the activity has yet to be decided. In our implementation of the SWO (described in Section 3.4), only one activity can be partially scheduled at any time.

3.3.1 Estimating the Utility of Unary Preferences. There are four types of unary preferences: duration preference, temporal domain preference, minimum distance preference, and maximum distance preference. Finding the maximum possible utility received by all of them simultaneously for a single activity is a computationally expensive problem, since numerous alternative plans need to be evaluated. To alleviate this problem, we make the simplifying assumption that the sum of the utility accumulated by the four alternative unary preferences is maximized by a plan that maximizes the utility received by a single unary preference. So, we try to maximize each source of utility separately by examining suitable plans; however, while trying to maximize a single source of utility, the utility resulting from the rest of the unary preferences is estimated as well. Our assumption is quite reasonable for linear models; however, the proposed model, although comprising several linear components, is not linear in general (e.g., temporal domains are not continuous), so our approach does not guarantee that the maximum possible utility received simultaneously by the four unary preferences will be found. The next paragraphs present the details of this approach.

Scheduling more duration for a single activity T_i (with a minimum of dur_i^{\min}) is not always the best option, since the extra duration could be scheduled at less preferred areas of the temporal domain, thus decreasing the utility received from domain preference defined by (3). Furthermore, more duration might reduce the utility received from proximity preferences, defined by (4) and (5). In general, extra duration makes the attempt to satisfy proximity preferences harder. So, in all cases when we attempt to estimate the utility received from unary preferences, we do it both for the minimum and the maximum possible duration of each activity.

Concerning domain preferences using Eq. (3), it is attempted to schedule each activity (except for those that have constant domain preference) as early as possible or as late as possible, according to the type of the domain preference. An approach similar to metric m_3 is followed, with parts having the maximum possible duration being scheduled one after the other, leaving the distance $dmin_i$ between them within the same domain interval.

Estimating utility that results from a unary minimum distance preference $pdmin_i$, we compute $diff(T_i)$ assuming various values d of the minimum distance constraint, such that $dmin_i \leq d \leq pdmin_i$, starting from higher values. As soon as we find a value of d for which $diff(T_i) \leq 1$, we estimate $U_i^{dmin}(\pi_i)$ of Eq. (4), using as π_i the first or the last (depending on the temporal domain

preference) scheduled parts of the activity (using the m_3 approach to construct π_i), up to the activity's duration.

Estimating $U_i^{dmax}(\pi_i)$ is more expensive. Indeed, $U_i^{dmax}(\pi_i)$ is maximized if all the parts of an interruptible activity are scheduled as close to each other as possible, i.e., at a distance equal to $dmin_i$. However, depending on the shape of the activity's temporal domain, $U_i^{dmax}(\pi_i)$ might depend on where in the activity's domain the earliest part will be scheduled. So, attempts to schedule the earliest part at different starting points within the activity's domain are needed, in order to find the maximum value for this type of utility. Our approach is to consider as starting point the left end of any interval of the activity's domain. Again, the m_3 approach is adopted to construct candidate plans. As soon as a plan π_i for the entire activity within $pdmax_i$ time is found, the attempts to maximize maximum distance utility are stopped.

For partially scheduled activities, the scheduled parts are also considered when estimating unary preferences. In particular, for domain preferences the scheduled parts contribute a fixed amount of utility, according to Eq. (3); for minimum distance preferences, a minimum distance d is maintained between the unscheduled and the scheduled parts of the activity, whereas the utility resulting from the already scheduled parts is also assessed. Finally, for maximum distance preferences, the distance between the new and the already scheduled parts, as well as the distance among the already scheduled parts, is taken into account.

Example 4. Suppose an interruptible activity T_i , with the following attributes: $dur_i^{\min} = dur_i^{\max} = 10$, $D_i = [0, 10] \cup [20, 35]$, linear descending domain preference function (similar to that of Figure 2(a)), $smin_i = 2$, $smax_i = 4$, $dmin_i = 3$, $dmax_i = \infty$, $pdmin_i = 10$ and $pdmax_i = 15$.

Trying to schedule this activity as early as possible (according to the m_3 heuristic), in order to maximize the linear descending temporal domain preference utility, results in the plan $\pi_1 = 0 : 4, 7 : 3, 20 : 3$, with $x:y$ denoting that a part with duration y has been scheduled at time x . Trying to maximize the minimum distance preference, we start with $d = pdmin_i = 10$, which results in $\pi_2 = 0 : 4, 20 : 4$, a plan with totally scheduled duration $dur_i = 8 < dur_i^{\min}$. Indeed, another part cannot start at $t = 34$, since the duration of this part would be at most 1, that is, violating the $smin_i$ constraint without furthermore accumulating at least dur_i^{\min} duration. Next we try $d = 9$, resulting in $\pi_3 = 0 : 4, 20 : 4, 33 : 2$, which is a valid plan. Finally, trying to maximize the maximum distance preference $pdmax_i$, we attempt creating two plans for T_i , one starting from 0 and another starting from 20, both having the minimum possible distance $dmin_i$ between their parts. The first is $\pi_4 = \pi_1$, whereas there is no plan starting from $t = 20$ with $dur_i \geq dur_i^{\min}$.

Plans π_1 , π_3 and π_4 are evaluated in terms of *all* unary preferences, that is, using Eqs. (2), (3), (4), and (5), with the best of them giving the estimate for the maximum utility that might result from T_i 's unary preferences. Note finally that in this example we considered a fixed duration for T_i . If $dur_i^{\min} < dur_i^{\max}$,

candidate plans should be formed both for $dur_i = dur_i^{\min}$ and for $dur_i = dur_i^{\max}$, thus resulting in about twice plans to evaluate.

3.3.2 Estimating the Utility of Binary Preferences. Recall from Section 2.2.2 that there are four types of binary preferences (soft constraints) that might hold between any pair of activities. Maximizing the total utility received from all of them simultaneously is a computationally expensive problem, requiring evaluation of numerous alternative pairs of plans for them. To alleviate this problem, we adopt two simplifying assumptions: First, the total utility received from all binary preferences over a pair of activities is maximized by a pair of plans that maximizes a single preference too. Second, binary constraints over the same pair of activities are ignored. These assumptions concern only temporal binary preferences, whose utility depends on the particular plans of the involved activities. On the other hand, implication preferences depend only on whether the activities are schedulable (or have already been scheduled) or not, so they are treated separately. In the next paragraphs we exemplify the pairs of plans that are evaluated for any pair of activities over which binary preferences hold. For any such pair of plans, all the binary preferences that hold over the two activities, as well as the unary preferences over the involved activities, are taken into account.

Concerning an ordering preference of the form $T_i < T_j$, the obvious way to get the maximum possible utility is to try to schedule T_i as early as possible and T_j as late as possible within their temporal domains (in a way similar to metric m_3) and thus a single pair of plans needs to be evaluated. Hence, parts of maximum possible duration are scheduled from left to right for T_i and from right to left for T_j , up to the maximum duration allowed for each activity. Then, the degree of satisfaction, and thus the resulting utilities, for the ordering preference is computed, using Eq. (7). Furthermore, in the presence of proximity preferences as well, additional utility might result from them. Finally, to increase accuracy, the utility resulting from the unary preferences of the two activities, using Eqs. (2) through (5), is also accumulated.

Minimum distance preferences are treated in a similar way. In order to minimize the distance between two activities T_i and T_j , it is sufficient to check two pairs of plans: one where T_i is scheduled as earlier as possible and T_j is scheduled as latest as possible, and another where T_j is scheduled as earlier as possible and T_i is scheduled as latest as possible, within their temporal domains in all cases. Again, Eqs. (2) to (5) and (7) are used to evaluate all unary and binary preferences over T_i and T_j .

Maximizing the utility resulting from binary maximum distance preferences is again a computationally expensive procedure. This utility is maximized whenever both activities are scheduled as compact as possible, that is, with minimum possible distances between their parts, and as close to each other as possible within their temporal domains. So, the degree of satisfaction depends on the start point of the plan of each involved activity, thus making the number of pairs of plans that need to be checked very high. In order to reduce the number of start point combinations to be checked, we perform two relaxations. First, we consider as candidate start points the left ends of the intervals of

their temporal domains only, with each candidate start point being used for the other activity as well if it falls within its domain. Second, each pair of candidate start points (one start point for each activity) is checked against some eligibility rules, in order for the respective plans to be evaluated. Actually, a case where T_i ends before T_j starts, when either T_i can start later while still ending before T_j starts or T_j can start earlier but not earlier than T_i ends, need not be considered. Again, Eqs. (2) through (5) and (7) are used to evaluate the overall utility received by unary and binary preferences over T_i and T_j for any pair of plans resulting from the eligible pairs of start points.

Evaluation of the various pairs of plans for pairs of activities T_i and T_j over which binary preferences exist is performed for up to four alternative duration combinations: $(dur_i^{\min}, dur_j^{\min})$, $(dur_i^{\max}, dur_j^{\min})$, $(dur_i^{\min}, dur_j^{\max})$ and $(dur_i^{\max}, dur_j^{\max})$, with identical pairs being ignored.

Finally, implication preferences are evaluated separately of the rest of the preferences, since they do not depend on particular plans but on the existence or not of any plan for the involved activities. So, an implication preference of the form $T_i \Rightarrow T_j$ yields all of its utility in all cases but the case where T_j cannot be scheduled (according to the difficulty measure) or has already decided not to be included in the plan (i.e., it is an activity earlier in the priority queue than the current activity) and T_i can be scheduled or has already been included in the plan.

Example 5. Suppose two interruptible activities, T_i and T_j , with the following attributes: $dur_i^{\min} = dur_i^{\max} = dur_j^{\min} = dur_j^{\max} = 10$, $D_i = [0, 20] \cup [30, 50]$, $D_j = [30, 50] \cup [60, 80]$, $smi_n_i = smi_n_j = 2$, $smax_i = smax_j = 4$, $dmi_n_i = dmi_n_j = 3$, $dmax_i = dmax_j = \infty$, an ordering preference $T_i < T_j$, a minimum distance preference $pdm_i_n_{ij} = 5$, and a maximum distance preference $pdmax_{ij} = 20$. No binary constraints are assumed between the two activities.

Based on the ordering preference, the pair of plans $\langle \pi_{i_1}, \pi_{j_1} \rangle$ will be evaluated, where $\pi_{i_1} = 0 : 4, 7 : 4, 14 : 2$ and $\pi_{j_1} = 64 : 2, 69 : 4, 76 : 4$, that is, T_i has been scheduled as early as possible and T_j has been scheduled as late as possible within their domains. Based on the minimum distance preference, in addition to the pair $\langle \pi_{i_1}, \pi_{j_1} \rangle$, the pair of plans $\langle \pi_{i_2}, \pi_{j_2} \rangle$ will be evaluated, where $\pi_{i_2} = 34 : 2, 39 : 4, 46 : 4$ and $\pi_{j_2} = 30 : 4, 37 : 4, 44 : 2$, that is, T_i has been scheduled as late as possible and T_j has been scheduled as early as possible (clearly, pair $\langle \pi_{i_2}, \pi_{j_2} \rangle$ is worse than $\langle \pi_{i_1}, \pi_{j_1} \rangle$ with respect to the minimum distance preference). Finally, based on the maximum distance preference, the following pairs will be initially considered.

$$\begin{aligned} \langle \pi_{i_3} = 0 : 4, 7 : 4, 14 : 2, \pi_{j_3} = 30 : 4, 37 : 4, 44 : 2 \rangle \\ \langle \pi_{i_4} = 0 : 4, 7 : 4, 14 : 2, \pi_{j_4} = 60 : 4, 67 : 4, 74 : 2 \rangle \\ \langle \pi_{i_5} = 30 : 4, 37 : 4, 44 : 2, \pi_{j_5} = 30 : 4, 37 : 4, 44 : 2 \rangle \\ \langle \pi_{i_6} = 30 : 4, 37 : 4, 44 : 2, \pi_{j_6} = 60 : 4, 67 : 4, 74 : 2 \rangle \end{aligned}$$

Pair $\langle \pi_{i_4}, \pi_{j_4} \rangle$ need not be considered, since there is an alternative pair of plans, that is, $\langle \pi_{i_6}, \pi_{j_6} \rangle$, where T_i starts later while still ending before T_j starts. So, pairs $\langle \pi_{i_1}, \pi_{j_1} \rangle$, $\langle \pi_{i_2}, \pi_{j_2} \rangle$, $\langle \pi_{i_3}, \pi_{j_3} \rangle$, $\langle \pi_{i_5}, \pi_{j_5} \rangle$ and $\langle \pi_{i_6}, \pi_{j_6} \rangle$ will be evaluated

in terms of all unary and binary preferences, that is, aggregating utilities from Eqs. (2) to (5) and (7), with the maximum sum characterizing the pair of activities T_i and T_j in terms of binary preferences. In this example we considered fixed durations for T_i and T_j . If $dur_i^{\min} < dur_i^{\max}$ or $dur_j^{\min} < dur_j^{\max}$, pairs of plans should be formed for all combinations of minimum and maximum durations of the two activities.

3.3.3 Aggregating Utilities. Estimates of the overall utility are needed each time a decision is to be taken. Decisions usually involve a subset of the undecided constraint variables, so for each alternative vector of values for a set of such variables, an estimate of the overall utility that will result after deciding the remaining constraint variables is needed as well. This overall estimate is based on the estimates of the utilities resulting from unary and binary preferences, so an aggregation method is needed. The approach we adopt is a combination of *sum* and *max* aggregation operators [Refanidis and Yorke-Smith 2009]. In particular, we assign a utility value to each activity using the *max* operator over the alternative sources of utility, and then we estimate the utility of the entire plan using the *sum* operator over all activities.

Specifically, for each activity T_i , an estimate \hat{u}_i of the maximum possible utility received from the combined unary preferences, according to Section 3.3.1, is computed. Furthermore, for any other activity T_j , such that binary preferences between T_i and T_j have been expressed, an estimate \hat{u}_{ij} of the maximum possible utility received from the combined binary preferences, according to Section 3.3.2, is also computed. Then, the total utility \hat{U}_i assigned to activity T_i is estimated by the following equation.

$$\hat{U}_i = \max \left(\hat{u}_i, \max_j \frac{\hat{u}_{ij}}{2} \right) \quad (10)$$

Finally, the overall utility of the plan \hat{U} , after deciding all the remaining constraint variables, is estimated by the following equation.

$$\hat{U} = \sum_{i=1}^N \hat{U}_i + \sum_{\substack{T_j \text{ in plan} \vee \\ T_i \text{ not in plan}}} u_{T_i \Rightarrow T_j} \quad (11)$$

Eq. (11) takes into account also the implication preferences, which are independent of the particular plans. So, for any implication preference $T_i \Rightarrow T_j$, the quantity $u_{T_i \Rightarrow T_j}$ is added to the overall utility in case T_j is schedulable (i.e., $diff(T_j) \leq 1$) or is already scheduled, or T_i is not schedulable (i.e., $diff(T_i) > 1$) or has already decided not to be scheduled.

3.4 Greedy Solution Construction

As described in Section 3.1, the basic SWO solution cycle comprises three phases: construct a solution, analyze it, and reorder the activities in the priority queue. The whole procedure is preceded by the initialization of the priority queue.

Although the initial ordering of activities in the priority queue is not crucial, a judicious ordering may save several iterations of the SWO cycle. Since our goal is to maximize the utility received from scheduling the activities, the initialization fills the priority queue in decreasing order of the maximum utility that can be produced by each activity, according to Eq. (10).

The most important step of the SWO solution cycle is the greedy construction of a solution. For each activity, a combined decision has to be taken as of when to schedule it, where to schedule it, and what amount of duration to allocate (in case of no fixed duration). For interruptible activities, these decisions have to be taken also for each part. Activities are scheduled one after the other, according to the order in which they appear in the priority queue. Failing to schedule an activity T_i results in not attempting to schedule any subsequent activity T_j in the presence of an implication constraint $T_j \Rightarrow T_i$. Forward checking (described in Section 3.7) occurs before scheduling the first activity, as well as after scheduling each activity. It occurs also when evaluating each alternative scheduling decision, in an attempt to improve the precision of the utility estimates.

In particular, in order to schedule a noninterruptible activity T_i , the combinations of the possible durations $dur_{i_1} \in [dur_i^{\min}, dur_i^{\max}]$, all alternative locations $l_{i_1} \in Loc_i$, and all possible start times t_{i_1} within its current temporal domain must be calculated. For each such combination, and following forward checking, the overall utility \hat{U} that is expected to result after scheduling all the remaining activities is computed, according to Eq. (11), and the best combination is adopted. Any combination that renders unschedulable, according to the difficulty metric, any subsequent activity T_j , for which an implication constraint $T_i \Rightarrow T_j$ exists, is ignored. In case no valid combination exists, T_i cannot be scheduled.

The procedure is similar for interruptible activities, as if each part of them was a separate activity. Parts are scheduled one after the other, with no backtracking allowed between them. Specifically, for each part T_{ij} , all the combinations of the possible durations $dur_{ij} \in [smin_i, smax_i]$, all alternative locations $l_{ij} \in Loc_i$, and all possible start times t_{ij} within the current temporal domain of T_i are evaluated, according to Eq. (11) and, following forward checking, the best combination is adopted. Again, any scheduling option that renders unschedulable either the remaining duration of the current activity, or any activity T_k later in the priority queue, such that an implication constraint $T_i \Rightarrow T_k$ exists, is ignored. In case that no valid combination for T_{ij} exists, while the required minimum duration dur_i^{\min} has not yet been scheduled, T_i cannot be scheduled and all previous parts of this activity are retracted from the current plan. Similarly, the consequences of forward checking while scheduling the previous parts of T_i are retracted from the temporal domains of the subsequent activities.

Parts of an interruptible activity are scheduled up to the maximum possible duration dur_i^{\max} ; however, as soon as the minimum required duration dur_i^{\min} has been achieved, there is the option not to schedule another part even if this is possible. The decision to schedule additional parts of the same interruptible activity beyond dur_i^{\min} is taken by computing estimates of the utility \hat{U} that

is expected to be received without and with scheduling them; if the former exceeds the latter, an additional part is not scheduled.

In any case of failing scheduling an activity T_i , all already scheduled activities T_j such that an implication constraint $T_j \Rightarrow T_i$ exists, are also retracted of the plan. The consequences of forward checking while scheduling any such activity T_j are also retracted from the temporal domains of the subsequent activities.

It is worth noting that, according to the previous paragraphs, there is no decision specifically taken for the number of parts p_i in which an interruptible activity T_i is decomposed. Besides, the proposed model does not allow expressing preferences either over the number of parts p_i or on their durations. So, what happens is that for each part T_{ij} , a combined decision is taken for $\langle dur_{ij}, l_{ij}, t_{ij} \rangle$, while trying to maximize Eq. (11). The number of parts p_i results as a consequence of the decisions on dur_{ij} .

3.5 Computing the Feasible Part Durations for Interruptible Activities

For an interruptible activity, depending on the values of dur_i^{\min} , dur_i^{\max} , smi_n_i , and $smax_i$, several durations for dur_{ij} may be prohibited. For example, if $dur_i^{\max} = 12$, $smax_i = 8$ and $smi_n_i = 5$, it is not possible to schedule a part of duration 8, since in this case the next part to be scheduled should have a maximum duration of 4, thus violating the constraint imposed by smi_n_i . We see that special reasoning is needed when allocating durations to the parts of an interruptible activity.

Identifying the feasible part durations to be scheduled preserves computation time, since not all possible part durations need be evaluated. However, there are situations that are inherently problematic, that is, no solution exists that satisfies all duration constraints. Consider the case $dur_i^{\min} = 14$, $dur_i^{\max} = 18$, $smi_n_i = 10$ and $smax_i = 12$, in which there is no combination of parts durations that sums between 14 and 18. For such cases, we decided to allow exceeding dur_i^{\max} to the least possible extent.

In order to check whether dur_{ij} is a feasible duration for the next part T_{ij} to be scheduled, we observe the following: Given smi_n_i and $smax_i$, the total durations dur_i that can be achieved while satisfying smi_n_i and $smax_i$ constraints are in $[K \cdot smi_n_i, K \cdot smax_i]$, for various values of the positive integer K . If the desired duration range, $[dur_i^{\min}, dur_i^{\max}]$ falls between two such intervals, that is, there is a value for K such that $K \cdot smax_i < dur_i^{\min}$ and $(K+1)smi_n_i > dur_i^{\max}$, then there is no solution.

So, given a remaining duration range to be scheduled, $[A, B]$, we identify the largest value of K , say K_1 , for which $K_1 \cdot smax_i < A$, and the smallest value of K , say K_2 , for which $K_2 \cdot smi_n_i > B$. K_1 and K_2 can be computed from the following equations.

$$K_1 = \lfloor (A - 1) / smax_i \rfloor \quad K_2 = \lceil (B + 1) / smi_n_i \rceil \quad (12)$$

If $K_2 = K_1 + 1$, then no solution exists, so dur_{ij} is not an acceptable duration for T_{ij} . In the case that no acceptable duration for T_{ij} can be found, then we relax dur_i^{\max} constraint to the smallest possible extent. This can be achieved by setting $dur_{ij} = smi_n_i$ for all remaining parts T_{ij} of T_i .

Example 6. Suppose $smin_i = 10$, $smax_i = 12$, and $dur_i^{\min} = 25$ and $dur_i^{\max} = 31$. Trying $dur_{i_1} = 12$ for T_{i_1} does not work. Indeed, $A = 13$, $B = 19$, $K_1 = \lfloor 12/12 \rfloor = 1$ and $K_2 = \lceil 20/10 \rceil = 2$. Trying $dur_{i_1} = 11$ works, since $A = 14$, $B = 20$, $K_1 = \lfloor 13/12 \rfloor = 1$ and $K_2 = \lceil 21/10 \rceil = 3$. Similarly, for $dur_{i_1} = 10$ we have since $A = 15$, $B = 21$, $K_1 = \lfloor 14/12 \rfloor = 1$ and $K_2 = \lceil 22/10 \rceil = 3$, that is, another feasible option. Supposing that $dur_{i_1} = 10$ is selected, the remaining duration is between 15 and 21. In the same way we can proceed with T_{i_2} and dur_{i_2} , finding that $dur_{i_2} = 12$ is not a feasible option, whereas $dur_{i_2} = 11$ and $dur_{i_2} = 10$ are, and so on, up to scheduling at least dur_i^{\min} duration.

3.6 Analyze, Reordering and Memoization

Before scheduling each activity T_i , Eq. (10) is applied to each subsequent activity T_j in the priority queue in order to estimate the utility \hat{U}_j that will result from scheduling each one of them. After having scheduled T_i , Eq. (10) is applied again to all subsequent activities to compute updated values \hat{U}'_j . Any subsequent activity T_j such that $\hat{U}'_j < \hat{U}_j$ is marked for promotion before T_i .

Promotion does not occur right after the identification of the flaw, but at the end of the current SWO iteration. The greedy solution construction continues with the current order, and by the end of the current SWO iteration all activities marked for promotion are promoted. In case an activity T_k was marked twice for promotion, before T_i and before T_j , with T_i being before T_j in the current priority queue, T_k is promoted before T_i . In case T_j was marked for promotion before T_i and T_k was marked for promotion before T_j , then in the new priority queue T_k will precede T_j , which in turn will precede T_i . In the extreme case where all activities were marked for promotion in all cases, the new priority queue will be the reverse of the old one.

There are two stopping criteria for the SWO cycles. The first criterion is to set a limit on the number of cycles. We adopted as a limit the identification of three consequent SWO iterations without any improvement on the utility U of the best solution found so far, according to Eq. (8). The second criterion takes into account the reordering mechanism. In case no reordering of the priority queue takes place after an SWO iteration, then iterations stop.

In order to avoid reevaluating the same order of the activities in the priority queue, a memoization procedure is needed. The full ordering of the activities in the priority queue is memoized before each SWO iteration. After reordering, the new order is checked against the memoized orders. In case a match is found, the new order is modified by cyclic right-shifting of all activities in the queue, that is, the first activity becomes second and the last activity becomes first. Right shift can be attempted up to N times consecutively; if no fresh order is found, SWO terminates.

3.7 Forward Checking

There are several constraints supported by the proposed model. Constraint propagation is a useful technique, since it can result in better estimates of the difficulty metrics and the utility estimates. However, enforcing arc consistency is not possible in this problem formulation because, with the exception of the

implication constraints, all other binary constraints are conditioned on the inclusion of the involved activities in the plan. Suppose, for example, an ordering constraints $T_i < T_j$ between two as yet undecided activities. We cannot use bounds consistency to rule out parts of their temporal domains, unless we are sure that both activities will be included in the plan. Only in the presence of an implication constraint of the form $T_i \Rightarrow T_j$, bounds consistency could be used to rule out parts of T_i 's temporal domain only.

The approach we adopted is to employ forward checking, that is, prune the temporal domains of the subsequent activities in the priority queue each time a decision is taken for the current activity. Forward checking is also employed when evaluating alternative decisions for the current activity. Specifically, for each alternative decision (see Section 3.4), the domains of the remaining activities are temporarily updated using forward checking and, thereafter, utility estimates are obtained. Full arc consistency is employed only for the implication constraints, both after taking a decision and each time a candidate decision is evaluating.

4. EMPIRICAL EVALUATION

Our empirical evaluation has two purposes. The first is to evaluate the proposed model in terms of effectiveness and efficiency. With *effectiveness* we refer to the quality of the resulting plans whereas with *efficiency* we refer to the time needed to find them. Since it is impractical to obtain optimal plans, it is difficult to precisely assess the quality of the resulting plans. To overcome this problem, we use theoretical upper bounds of the optimal plans to compare with. Further, we couple the adopted SWO method with a local-search postoptimization procedure, trying to further improve the resulting plans. The second purpose is to get insights of how the adopted SWO method works in practice, to ascertain whether this framework was a fitting choice for the specific problem or not.

We implemented the algorithms described in Section 3 and we created several problem instances, using a random problem generator employing uniform distributions for the various choices. The problems generated differ only in the number of activities, whereas the rest of the parameters are constant. The planning horizon was set to 500 time units, with the temporal domain of each activity being a random set of intervals covering the entire planning horizon. 70% of the activities had fixed duration. 40% of the activities were interruptible with minimum distance constraints, 30% of these had maximum distance constraints, 30% had minimum distance preferences, and 30% had maximum distance preference. 20% of the activities had utilization 0.5; the rest had full utilization. Binary constraints and preferences were defined with probability $1/(2N)$ for every pair of activities and every type of constraint or preference. We defined 4 locations, with random subsets of them assigned to the activities. One of the locations had zero distance to every other, thus corresponding to an *anywhere* location.

Activity utilities were selected from the interval [5,12], temporal preference utilities from the interval [5,10], and duration utilities from the interval [2,5]. All utilities assigned to binary preferences were selected from the interval [1,3].

In our design choices we tried to simulate a real calendar with 30-minutes time slot. Accordingly, the time horizon corresponds to about 10 days, whereas the intervals of the temporal domain had an average width of 20 time units (i.e., 10 hours). The problem instances can be considered difficult since all activities are fluent in time, have large domains, and many types of constraints and preferences are imposed on them. Our experience with the SELFPLANNER system indicates that usually activities are much simpler than in our test suite.

The reported results focus both on the quality of the resulting plans, as well on the time needed to solve them. As already mentioned, it is impractical to obtain optimal solutions. Hence, we compare the quality of the solutions produced by the proposed methods to a *theoretical* upper bound, which can be obtained by summing up the maximum values for all alternative sources of utility. Of course, the more the activities and the binary constraints and preferences over them, the looser this theoretical upper bound.

As an additional indication, we developed a simple but powerful postoptimization procedure to further improve the plans produced by the SWO method. In particular, after the termination of all SWO iterations, postoptimization tries to further improve the best plan found by attempting to move any scheduled part of any activity anywhere else within its domain. Provided that the new plan is valid, each such change is evaluated according to Eqs. (2) to (5) and (7) and it is adopted in case of a utility increase. These postoptimization attempts continue repeatedly until no further improvement is possible. The postoptimization procedure does not change the durations of the scheduled parts or their locations, and does not attempt to schedule any unscheduled activity.

Table I presents the results. We have created five random problem instances of several sizes between 5 and 60 activities. The experiments were performed on an AMD Turion TL-60 64-bit machine running at 2 GHz, with 3GB memory, running MS Windows 7; the code was written in C++, it was compiled using MS-Visual Studio 8 with /Ox full optimization and was run out of the development platform. Figure 5 depicts graphically the relation between the average quality (after postoptimization) and time to the problem size.²

As can be seen from the results, the average quality (after postoptimization) of the resulting plans falls with the number of activities, starting from 94% of the theoretical upper bound for 5 activities and falling to 77% for 60 activities. The correlation coefficient of a linear regression is strongly negative ($r = -0.90$). The trend of quality decrease with problems with more activities was expected. Indeed, the greater the number of activities, the greater the number of constraints and preferences between them, hence the theoretically computed upper bound is expected to be looser. Suppose, for example, that activity T_i is preferred to be scheduled as early as possible, whereas activity T_j is preferred to be scheduled as late as possible. The existence of an ordering constraint or preference $T_j < T_i$ or a maximum distance constraint or preference between them renders it impossible to get the maximum possible utility from all these sources of utility. Furthermore, for problems with many activities, some of the activities cannot be scheduled due to the hard constraints (the time

²The code and the problem files are available from <http://eos.uom.gr/~yrefanid/tist2010>.

Table I. Experimental Results

#	N	Iter	Time (secs)	Quality % (without/with post-optm)	Not scheduled	#	N	Iter	Time (secs)	Quality % (without/with post-optm)	Not scheduled
p5_1	5	1	0.5	99.44/99.50	0	p35_1	35	4	27.5	85.01/85.05	0
p5_2	5	4	3.6	97.02	0	p35_2	35	8	44.0	89.56/90.16	0
p5_3	5	5	2.3	93.02/93.10	0	p35_3	35	5	16.3	80.47/81.01	2
p5_4	5	2	16.3	93.34	0	p35_4	35	8	51.2	84.89/85.10	3
p5_5	5	4	4.8	86.33	0	p35_5	35	8	29.8	89.39/89.61	0
p10_1	10	7	13.9	84.85/85.39	0	p40_1	40	4	23.1	83.70/84.74	1
p10_2	10	6	10.5	85.50/86.25	0	p40_2	40	7	37.9	81.23/81.32	3
p10_3	10	4	11.4	86.75/89.60	0	p40_3	40	9	48.6	87.65/87.88	0
p10_4	10	5	10.3	89.36/90.40	0	p40_4	40	4	18.6	75.39/76.09	5
p10_5	10	5	25.5	92.03/92.22	0	p40_5	40	8	54.3	85.43/85.46	0
p15_1	15	6	8.7	91.51	0	p45_1	45	6	43.7	80.68/82.11	3
p15_2	15	4	6.3	83.10/83.22	0	p45_2	45	4	25.3	83.44/83.52	2
p15_3	15	4	5.0	86.09/86.98	0	p45_3	45	8	51.8	90.05/90.49	0
p15_4	15	4	9.0	83.69/84.59	0	p45_4	45	6	42.0	79.11/79.13	4
p15_5	15	7	18.8	90.34/90.39	0	p45_5	45	7	48.0	87.46/87.72	1
p20_1	20	5	22.0	90.40/90.95	0	p50_1	50	6	51.7	83.95/84.22	0
p20_2	20	4	10.8	81.06/83.48	1	p50_2	50	10	92.6	85.63/85.83	1
p20_3	20	5	29.3	89.54/90.09	0	p50_3	50	6	52.5	82.21/82.64	3
p20_4	20	8	25.3	86.14/86.90	0	p50_4	50	7	71.8	83.35/83.44	2
p20_5	20	4	18.6	92.13/93.56	0	p50_5	50	4	29.2	79.43/79.98	2
p25_1	25	4	17.7	87.14/89.71	0	p55_1	55	6	61.9	82.98/83.13	3
p25_2	25	8	34.7	89.74/90.20	0	p55_2	55	4	32.5	76.38/76.61	5
p25_3	25	6	22.1	90.71	0	p55_3	55	4	35.3	77.69/78.02	3
p25_4	25	4	11.7	88.91/91.07	1	p55_4	55	6	46.7	77.03/77.57	5
p25_5	25	8	28.6	92.94/93.77	0	p55_5	55	6	58.6	84.25/85.38	3
p30_1	30	10	46.1	89.48/89.92	0	p60_1	60	8	91.3	76.56/76.57	8
p30_2	30	6	20.2	90.58/90.75	0	p60_2	60	5	52.3	78.14/78.39	6
p30_3	30	4	27.1	85.72/86.23	1	p60_3	60	7	73.9	73.14/73.23	8
p30_4	30	7	28.6	89.85	0	p60_4	60	10	87.6	81.12/82.05	3
p30_5	30	7	35.6	89.43	0	p60_5	60	7	102.4	74.46/74.67	7

For each problem instance (#) we show: the number of activities N; the number of SWO cycles performed (Iter); the time needed (Time); the quality of the resulting plan without/with postoptimization as a percentage of the theoretical upper bound (Quality %); and the number of the not scheduled activities.

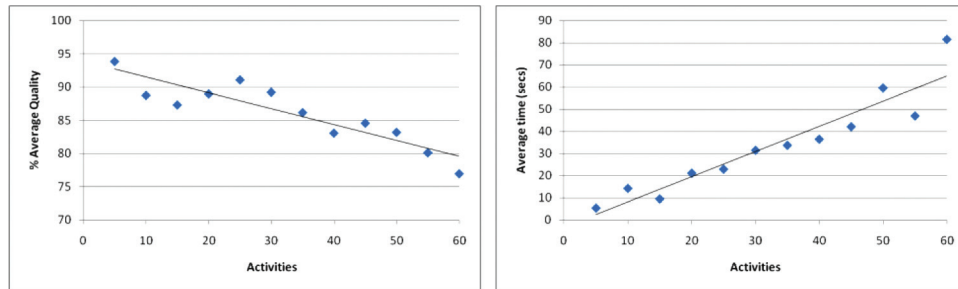


Fig. 5. (a) Average quality (after postoptimization) as a function of the number of activities. (b) Average solution time as a function of the number of activities.

horizon is the same in all cases). Hence, the utility these activities might have contributed (which is taken into account in the theoretical upper bound) cannot be received. Finally, more activities render the optimization problem more difficult, so it is expected that the performance of any incomplete optimization algorithm decreases.

The time needed to solve the problems also increases with the number of activities, following a linear trend with a strong positive correlation coefficient ($r = 0.94$), as can be seen from Figure 5(b). Indeed, 5.5 seconds are needed on average to solve problems with 5 activities and 81 seconds for problems with 60 activities. These results demonstrate that, under the specific experimental settings, SWO scales up very well and hence is a suitable solving methodological choice to tackle our optimization problem. Of course, it does not provide any guaranty for the quality of the resulting plans; however, this is a known characteristic of the greedy algorithms and, as explained in the Introduction, it was a choice to adopt such an approach.

It is interesting to compare the quality of the resulting plans without and with postoptimization. Although there are cases where postoptimization does not bring anything at all (e.g., problems p5_2, p5_4, p5_5, etc.), in most cases there is some increase in the quality. The average increase over the whole set of problems is 0.62% (using the quality of the nonpostoptimized plan as a basis), which is noticeable but not significant. To further explore the impact of the postoptimization procedure to the overall approach, we rerun the SWO algorithm with employing the postoptimization procedure after each iteration (instead of once only, at the end of the iterations). This modification might affect both the evaluation of the intermediate plans constructed by SWO (step 3b in Figure 4), which should be evaluated after postoptimization, as well as the termination conditions of the main loop (step 3), since the main termination condition imposes termination after not encountering any better plan for three consecutive iterations.

After rerunning the SWO algorithm with postoptimization after each iteration, we encountered changes in the behavior (besides the increase in the planning time) in only six out of the sixty (i.e., 10%) problem instances. These changes are shown in Table II. As expected, in all other cases not shown in Table II, the time needed to produce the plans was increased.

Table II. Experimental Results When Using Postoptimization After Each Iteration with respect to Using Optimization Once Only at the End of All Iterations

	Post-optimization at the end only				Post-optimization after each iteration			
	Iter	Time (secs)	Quality %	Not scheduled	Iter	Time (secs)	Quality %	Not scheduled
p5.4	2	16.3	93.34	0	2	16.3	93.66	0
p10.5	5	25.5	92.22	0	4	19.6	92.78	0
p15.5	7	18.8	90.39	0	5	14.6	90.39	0
p20.4	8	25.3	86.90	0	10	33.3	86.93	0
p25.2	8	34.7	90.20	0	5	23.4	90.26	0
p55.4	6	46.7	77.57	5	6	49.4	77.58	5

For each problem instance (#) and for each mode we show: the number of SWO cycles performed (Iter); the time needed (Time); the quality of the resulting plan as a percentage of the theoretical upper bound (Quality %); and the number of the not scheduled activities.

Table III. Quality (as a percentage of the theoretical upper bound) of the Plans Generated After Each SWO Iteration for Problem p30.1, Without and with Postoptimization

Iteration	1	2	3	4	5	6	7	8	9	10
without post-optimization	82.90%	82.26%	86.67%	82.84%	88.51%	82.22%	89.49%	81.51%	87.21%	86.64%
with post-optimization	83.06%	83.45%	87.07%	82.90%	88.86%	83.08%	89.92%	81.86%	87.61%	87.38%

As can be seen from Table II, the increase in the quality by running postoptimization after each SWO iteration is negligible even in these six cases. This is another indication that the proposed SWO solver produces near-locally-optimal plans, at least under the transformations supported by the employed postoptimization procedure. Concerning the number of iterations in the six cases of Table II, in four cases there is a decrease, whereas in one case there is an increase. Finally, concerning the time needed to solve the problems, there is an increase of 5.25% when considering the whole set of problems (not shown in Table II), which is noticeable.

In order to get a better insight of what happens inside the SWO cycles, it is interesting to monitor one case. We selected problem p30.1, since it is the smallest problem instance with the largest number of iterations (i.e., 10) that we encountered in our problem set. Table III shows the percentage quality of the plans generated at each iteration of SWO for this problem instance, without and with postoptimization.

As can be seen, the best plan (according to Table I) with percentage quality equal to 89.92 was found at iteration 7. With the exceptions of iterations 4 and 6, the quality of the plans generated during the first seven iterations continuously increases. The solving halts after iteration 10, since there is no improvement in the last three iterations. The decrease in the quality both before and after the best plan has been found can partially be attributed to the memoization technique, which might prune good orderings that have been met before and replace them by a cyclic rotation of the priority queue. Finally, it is worth mentioning that the quality improvement due to the postoptimization procedure at the end of each iteration is not that significant compared to the

differences in the quality between the base plans found by SWO at each iteration. This explains why for the vast majority of the problems nothing has changed by introducing postoptimization after each iteration. It also serves as an additional evidence that the SWO algorithm, as has been adapted to this problem, was a good choice.

5. RELATED WORK

Palen [1999] notes some of the many ways that AI technologies can aid an individual's time management. Recent ambitious integrated AI efforts that include time management include Freed et al. [2008] and Myers et al. [2007]. We restrict attention to application of automated scheduling technologies. Modi et al. [2004] and Berry et al. [2009] survey pioneering efforts to schedule tasks, noting the importance of user interaction in order to develop satisfactory plans. A separate problem is assisting with task prioritization; one of the many such efforts is Varakantham and Smith [2007; 2008]. More closely related to the problem of our interest, but again distinct, is negotiation of meetings between multiple individuals; as noted in Berry et al. [2009], a series of assistive tools that employ AI have been developed for meeting scheduling assistance. A flavor of such tools is the system of Zabala et al. [2001], which takes events from an online todo list and schedules them; the preference model and the allowed constraints are simple.

Due to the individualism of the use and management of time [Payne 1993; Palen 1999], inherent in the domain are complex constraints, objectives, and preferences. These present challenges to AI planning and scheduling [Smith 2003]. Further, new and changing tasks and events make the domain dynamic [Gallagher et al. 2006; Verfaillie and Jussien 2005].

The SELFPLANNER system [Refanidis and Alexiadis 2008] takes events explicitly by the user and schedules them. The system implements a subset of the model we consider, by not supporting binary constraints and preferences (with the exception of the ordering constraints), intraactivity distance preferences, intraactivity maximum distance constraints, activity utilities, activity utilization, duration ranges, and alternative locations for each activity. Its model is described in detail by Refanidis [2007], who also describes the solving approach based on squeaky wheel optimization. A more expressive, but informal model formulation from which we draw is described by Refanidis and Yorke-Smith [2009], who, however, do not present means to solve their model.

Expressive constraints and powerful propagation techniques, notably for global constraints, account for the known success of constraint-based approaches to scheduling [Baptiste et al. 2001]. Laborie [2009] presents generic models for three diverse scheduling problems using the commercial IBM ILOG Scheduler tool. Among the three problems is the personal task scheduling model of Refanidis [2007]. Laborie's work improves over Refanidis [2007] in terms of the quality of the resulting plans. However, solution times per problem are not reported, whereas the domain tackled by both Refanidis [2007] and Laborie [2009] is significantly less expressive than our model.

Squeaky wheel optimization, since its introduction by Joslin and Clements [1999] and application to a manufacturing problem, has been successfully applied to a range of scheduling problems, including task allocation in over-subscribed situations [Kramer and Smith 2003], planning for robot activities [Joslin et al. 2005], personnel scheduling [Lim et al. 2003], and machine scheduling [Feng and Lau 2008]. Aickelin et al. [2009] apply a variant of SWO, *evolutionary SWO*, to two personnel scheduling problems. Evolutionary SWO adds selection and mutation in an attempt to increase the evolution of the solution.

6. CONCLUSION AND FUTURE WORK

In this article we developed a rigorous model for a rich variety of activities of an individual, accounting for intra- and interactivity constraints and preferences. We presented a solution approach, based on the well-known squeaky wheel optimization framework, enhanced with domain-dependent heuristics and forward checking, to cope with the underlying optimization problem. Empirical evaluation demonstrates that the approach is both effective, producing qualitative plans, and efficient, solving problem instances rapidly and scaling readily with problem size.

The present work can be extended in several directions. First, from an algorithmic point of view, we seek to further increase the quality of the resulting plans by designing more effective heuristics or coupling SWO with an enhanced local-search postoptimization procedure taking into account all the decision variables. In the same direction, we are investigating ways to further decrease the solution time by, for example, heuristically selecting a subset of scheduling options to evaluate, that is, when and where to schedule an activity and how much duration to allocate to it. We are interested in evaluating our algorithms by devising tighter theoretical upper bounds for the resulting utility, or comparing the SWO framework to other constraint optimization frameworks.

Second, from a modeling point of view, we want to examine extending the model in order to accommodate other aspects of an individual's activities, for example, nonmonotonic temporal preferences, additional types of constraints and preferences, or other forms of the degree of satisfaction for binary preferences. Any change in the model is expected to require additional work in the algorithmic dimension.

Finally, from an application point of view, it is in our immediate plans to incorporate the new model and solver to the SELFPLANNER system and obtain feedback from the actual users. In this integration, several engineering issues concerning the system's usability need to be addressed. A long term goal is to integrate scheduling an individual's activities with scheduling joint activities between different users of an intelligent calendar application.

ACKNOWLEDGMENTS

The authors express thanks to P. Berry and B. Peintner, and to the anonymous reviewers for their constructive suggestions.

REFERENCES

- AICKELIN, U., BURKE, E. K., AND LI, J. 2009. An evolutionary squeaky wheel optimization approach to personnel scheduling. *IEEE Trans. Evolut. Comput.* 13, 2, 433–443.
- ALEXIADIS, A. AND REFANIDIS, I. 2009. Defining a task's temporal domain for intelligent calendar applications. In *Proceedings of the 5th IFIP Conference on Artificial Intelligence Applications and Innovations (AIAD)*. L. Iliadis et al. Eds., Springer, 399–406.
- BAPTISTE, P., PAPE, C. L., AND NULJEN, W. 2001. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer, Boston, MA.
- BERRY, P. M., 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. In *Proceedings of the 21st Conference on Innovative Applications of Artificial Intelligence (IAAI)*. K. Haigh and N. Rychtyckjy Eds., AAAI Press, 40–46.
- FENG, G. AND LAU, H. C. 2008. Efficient algorithms for machine scheduling problems with earliness and tardiness penalties. *Ann. Oper. Res.* 159, 1, 83–95.
- FREED, M., CARBONELL, J., GORDON, G., HAYES, J., MYERS, B., SIEWIOREK, D., SMITH, S. F., STEINFELD, A., AND TOMASIC, A. 2008. RADAR: A personal assistant that learns to reduce email overload. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*. D. Fox and C. P. Gomes Eds., AAAI Press, 1287–1293.
- GALLAGHER, A., ZIMMERMAN, T. L., AND SMITH, S. F. 2006. Incremental scheduling to maximize quality in a dynamic environment. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS)*, D. Long, et al. Eds., AAAI Press, 222–232.
- JOSLIN, D. AND CLEMENTS, D. P. 1999. Squeaky wheel optimization. *J. Artif. Intell. Res.* 10, 365–397.
- JOSLIN, D., FRANK, J., JÓNSSON, A. K., AND SMITH, D. E. 2005. Simulation-Based planning for planetary rover experiments. In *Proceedings of the 37th Winter Simulation Conference*, M. E. Kuhl et al., Eds. 1049–1058.
- KRAMER, L. A. AND SMITH, S. F. 2003. Maximizing flexibility: A retraction heuristic for oversubscribed scheduling problems. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*. G. Gottlob and T. Walsh, Eds., Morgan Kaufmann, 1218–1223.
- LABORIE, P. 2009. IBM ILOG CP optimizer for detailed scheduling illustrated on three problems. In *Proceedings of the 6th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR)*. W.-J. van Hoeve and J. N. Hooker Eds., Springer, 148–162.
- LIM, A., RODRIGUES, B., AND SONG, L. 2003. Manpower scheduling with Time Windows. In *Proceedings of the 18th ACM Symposium on Applied Computing*. G. B. Lamont et al. Eds., ACM, New York, 741–746.
- MODI, P. J., VELOSO, M. M., SMITH, S. F., AND OH, J. 2004. CMRadar: A personal assistant agent for calendar management. In *Proceedings of Conference on Agent-Oriented Information Systems*. P. Giorgini et al. Eds., Springer, 169–181.
- MYERS, K., BERRY, P., BLYTHE, J., CONLEY, K., GERVASIO, M., MCGUINNESS, D., MORLEY, D., PFEFFER, A., POLLACK, M., AND TAMBE, M. 2007. An intelligent personal assistant for task and time management. *AI Mag.* 28, 2, 47–61.
- PALEN, L. 1999. Social, individual and technological issues for groupware calendar systems. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*. M. G. Williams and M. W. Altom Eds., ACM, New York, 17–24.
- PAYNE, S. J. 1993. Understanding calendar use. *Hum.-Comput. Interact.* 8, 2, 83–100.
- REFANIDIS, I. 2007. Managing personal tasks with time constraints and preferences. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS)*. M. Boddy et al. Eds., AAAI Press, 272–279.
- REFANIDIS, I. AND ALEXIADIS, A. 2008. SELFPLANNER: Planning your time! In *Proceedings of ICAPS Scheduling and Planning Applications Workshop (SPARK)*. L. Castillo et al. Eds.
- REFANIDIS, I., GEMITZIS D., AND STEPHANIDES, G. 2006. Scheduling personal time using squeaky wheel optimization. In *Proceedings of the ECAI Workshop on Modelling and Solving Problems with Constraints*. S. Prestwich and B. Hnich, Eds., 17–24.

- REFANIDIS, I., MCCLUSKEY, T. L., AND DIMOPOULOS, Y. 2004. Planning services for individuals: A new challenge for the planning community. In *Proceedings of the ICAPS Workshop on Connecting Planning Theory with Practice*. S. Biundo and P. Jarvis Eds., 56–61.
- REFANIDIS, I. AND YORKE-SMITH, N. 2009. On scheduling events and tasks by an intelligent calendar assistant. In *Proceedings of the ICAPS Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems*. M. A. Salido and R. Bartak Eds., 43–52.
- SMITH, S. F. 2003. Is scheduling a solved problem? In *Proceedings of the 1st Multi-Disciplinary International Conference on Scheduling: Theory and Applications (MISTA)*. G. Kendall et al. Eds., Springer, 3–18.
- VARAKANTHAM, P. AND SMITH, S. F. 2007. Linear relaxation techniques for task management in uncertain settings. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS)*. M. Boddy et al. Eds., AAAI Press, 272–279.
- VARAKANTHAM, P. AND SMITH, S. F. 2008. Advising busy users on how to cut corners in uncertain settings. In *Proceedings of the ICAPS Workshop on Oversubscribed Planning and Scheduling*. L. Barbulescu et al. Eds.
- VERFAILLIE, G. AND JUSSIEN, N. 2005. Constraint solving in uncertain and dynamic environments — A survey. *Constraints* 10, 3, 253–281.
- ZABALA, L., PERFECTO, C., UNZILLA, J., AND FERRO, A. 2001. Integrating automatic task scheduling and Web-based agenda in a virtual campus environment. In *Proceedings of the 2nd International Conference on Information Technology Based Higher Education and Training (ITHET)*. H. Akiyama, Ed.

Received March 2010; revised July 2010; accepted July 2010