SELFPLANNER: An Intelligent Web-based Calendar Application

Ioannis Refanidis and Anastasios Alexiadis

University of Macedonia, Dept. of Applied Informatics, Thessaloniki, Greece
54006, Thessaloniki, Greece
E-mails: yrefanid@uom.gr, talex@java.uom.gr

Abstract

This paper presents SELFPLANNER, a web-based intelligent calendar application that plans a user's tasks. The user enters her tasks along with task details, i.e. duration, release time and deadline, location, unary and binary constraints and preferences, and the application schedules the tasks and presents the resulting plan using Google's Calendar application. Whenever new tasks arrive, the user may ask for incremental replanning, whereas attention is paid to keep the original plan as unchanged as possible. SELFPLANNER supports both non-preemptive and preemptive tasks, with additional types of constraints over the preemptive ones. It also assigns location references to the tasks, whereas travel times are taken into account while scheduling. The user may impose ordering constraints among the tasks. Unary preferences assign utilities to the tasks' domains, whereas binary preferences concern minimizing or maximizing the distance between pairs of tasks. SELFPLANNER solves the planning problem using an adaptation of the Squeaky Wheel Optimization framework.

Introduction

Modern electronic organizers, such as MS-Outlook 2007, Google Calendar and Yahoo Calendar, do not provide for automatic scheduling of users' tasks. Users have to manually place their tasks into the calendar, as well as to arrange meetings with others. These applications provide various functionalities to assist the user to put her tasks into the calendar, detect conflicts, merge calendars, assign tasks to other users, arrange meetings, share and publish her calendar. The need for intelligent assistance to schedule a user's tasks has already been identified (Refanidis, McCluskey and Dimopoulos, 2004). It is a general impression that most of the effort in developing new editions of office applications is towards personal time management simplification. However, current status still remains behind automatic scheduling abilities (not to speak about planning) being incorporated into these programs.

This paper concerns an intelligent web-based calendar application, called SELFPLANNER, that automates task scheduling in the presence of time constraints and preferences. The user enters her tasks, together with a set of accompanying attributes for each task. In particular, each task is characterized by the following attributes:

- *Duration*: A reasonable upper bound for the overall duration of the task.
- *Domain*: The allowed time windows when the task can be scheduled.
- *Location*: The location where the user must be in order to perform the task.
- *Preemptive*: Whether the task is preemptive or not. Preemptive tasks can be executed in parts.
- *Part min duration*: For preemptive tasks, the minimum allowed duration for each part of it.
- *Part max duration*: For preemptive tasks, the maximum allowed duration for each part of it.
- *Part min distance*: For preemptive tasks, the minimum allowed temporal distance between each pair of parts of the task.
- *Unit preference*: A function over the task's domain that assigns utility values to each possible time unit where the task can be scheduled.

In addition, the following information constitutes part of the scheduling problem:

- *Travel times*: The time needed for the user to travel between every pair of locations.
- *Binary temporal constraints*: Some tasks must be executed before other tasks.
- *Binary preferences*: There may exist preferences over the distance between pairs of tasks, e.g. we might prefer two tasks to be scheduled as close (or as far) to each other as possible.

SELFPLANNER facilitates the entry of the task details. Perhaps the most tedious part of the data entry is the definition of the domain of each task. A domain may consist of several time intervals, distributed among many days/weeks. In order to facilitate domain definition, SELFPLANNER supports an implicit definition through the use of unary constraint templates. A constraint template is a set of allowed time windows with a predetermined makespan (e.g. a day or a week), without an absolute time reference. For example, a constraint template with a week's duration may define the office working hours. Another constraint template with a day's duration may define the sleeping time. Each constraint template may be used in two ways: Either positively, i.e. in order to include time slots into a task's domain, or negatively, in order to exclude time slots from a task's domain. SELFPLANNER provides several constraint templates, however the user may define her own ones. So, in order to define a task's domain, the user must provide a release date, a deadline, and combine several

constraint templates, either positively, other negatively, in order to include/exclude time windows.

SELFPLANNER is a web based application running on a dedicated web/planning server (Figure 1). All user's data are stored in the web/planning server, so the user can access SELFPLANNER from any networked computer. The user enters/edits task data using user-friendly forms. The web/planning server solves the scheduling problem and inserts suitable entries in the user's Google Calendar account, using Google's API. Finally, the user watches her calendar directly into Google Calendar. Currently SELFPLANNER does not support editing the calendar directly on Google's pages; however we plan to give this option in the future.

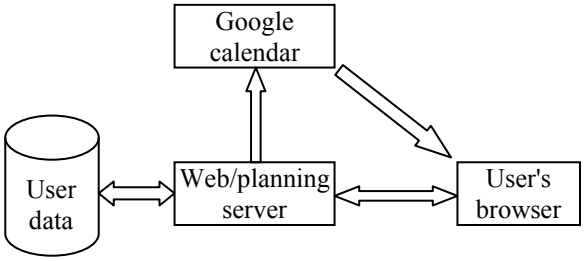

Figure 1: SELFPLANNER overall architecture

In order to solve the planning problem, SELFPLANNER uses a heuristic search algorithm based on an adaptation of the Squeaky Wheel Optimization framework (Joslin and Clements, 1999; Refanidis, 2007). The algorithm is incomplete, so it does not guarantee to solve the problem, even if a solution exists. However, extensive experimental results have shown that the algorithm is quite effective and efficient wrt complete search algorithms. The algorithm implemented in SELFPLANNER is suitable for incremental scheduling, i.e. tasks are scheduled as they arrive. Attention is paid to not incur significant changes to the existing schedule each time a new task arrives, however in case of tight schedules changes may be unavoidable.

SELFPLANNER has been developed in Java. In particular, the interface consists of a Java applet with a main window and several dialog boxes, whereas the core of the application runs on the web/planning server and is responsible for communicating with the user, the data base (mySQL) and Google calendar. The planning engine, i.e. the subroutine that solves the planning problem, has been developed in C++.

Use Case Scenario

After the necessary login phase, the main application window appears (Figure 2). This window displays the current list of task and supports five functions:

- Inserting a new task by clicking the `New Task` button.
- Editing an existing task by double clicking the task name in the list of tasks.
- Editing the table of locations and their mutual distances by clicking the button `Locs`.
- Editing the binary constraints and preferences by clicking the button `Prefs`.
- Editing several parameters of the SWO algorithm by clicking the button `Params`.

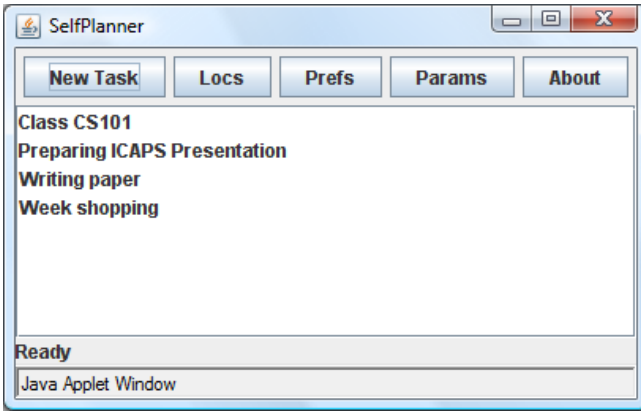

Figure 2: SELFPLANNER main window.

By clicking the `New Task` button/double clicking a task in the task list, the window `New/Edit Task` appears (Figure 3). In the `New Task` tab the user can enter the main task details, i.e. the task's name, duration, location and whether the task is preemptive (i.e. interruptible) or not. In case of interruptible tasks, the user has to enter the minimum and maximum duration for each part of the task, as well as the minimum temporal distance for each pair of parts of the task.

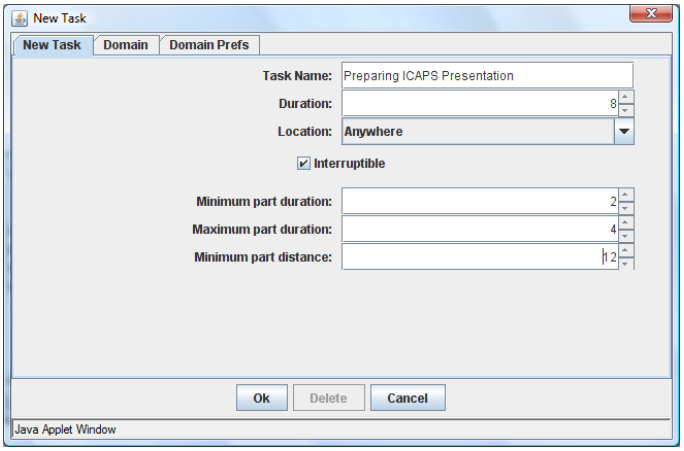

Figure 3: The `New Task` window.

In tab `Domain` the user has to define the domain of the task, i.e. when the task can be scheduled (Figure 4). Apart from determining the release time and the deadline for the task, the user may define the domain using a combination of general templates, such as weekly working hours, daily sleeping hours etc. Currently, SELFPLANNER provides a set

of predetermined templates; however we plan to give the user the possibility to create new templates. Of course the user has the possibility to define the domain by determining directly on a calendar the allowed time slots. Determining the domain of a task is the most critical part of SELFPLANNER interface, so we are working hard to make it as functional and user-friendly as possible.

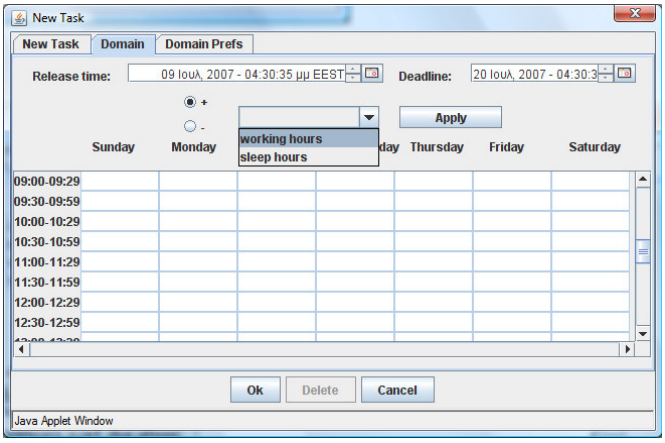

Figure 4: Determining a task's domain.

In tab `Preferences` the user can assign utilities in the various time slots of a task's domain (Figure 5). Currently, five different utility functions are supported: constant function, linear ascending/descending and step ascending/descending functions. Depending of the type of the utility function, the user has to enter the minimum and maximum utility of the domain, and the position of the step.

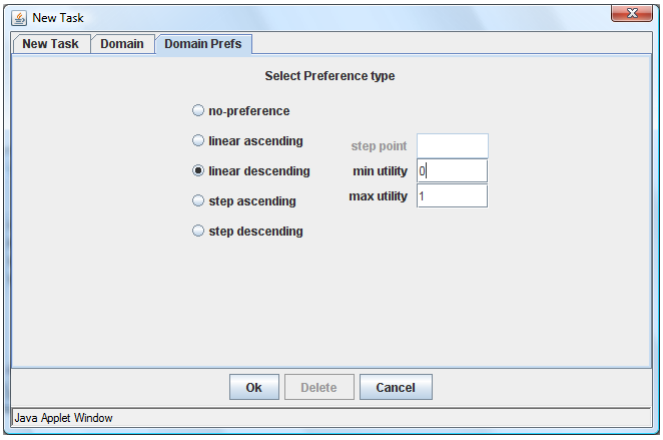

Figure 5: Defining unary preferences.

Figure 2 displays four tasks in the user's task list. By clicking the `Prefs` button the user can define binary constraint and preference relations between the tasks (Figure 6). Concerning constraints, currently only ordering constraints are supported. For example, in Figure 6 we show that tasks "Preparing ICAPS Presentation" and "Writing paper" have to be executed before task "Week shopping". Concerning binary preferences, only proximity preferences are supported. In particular, for any pair of tasks the user can state that the two tasks must be scheduled either as close or as far to each other as possible. For example, "Writing paper" should be executed as far as possible to the task "Class CS101", since it is expected that immediate before or after teaching a class is not the best period to think and write papers.

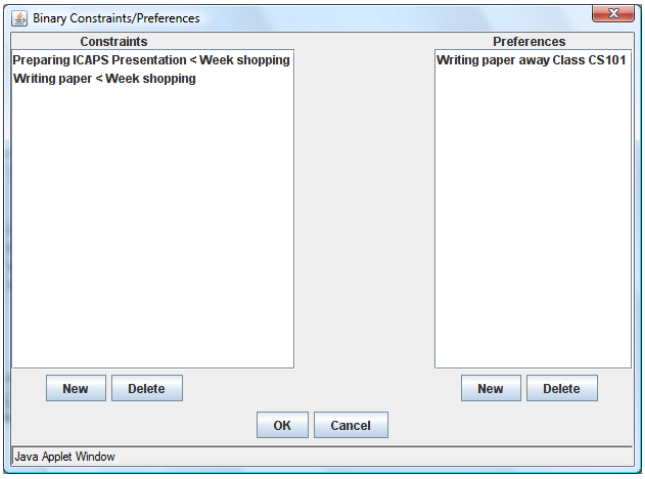

Figure 6: Defining binary constraints and preferences.

Each time the user enters a new task, SELFPLANNER solves the planning problem and presents the results directly on the user's account in Google calendar (Figure 7).

Conclusion and Future Work

SELFPLANNER is an ongoing work. The version described in this paper is an initial prototype with many limitations. Future extensions of SELFPLANNER include:

- *Alternative locations for each task*. Currently, a specific location is assigned to each task, whereas there is a special `anywhere` possibility (e.g. I can prepare a presentation anywhere, provided that I have my laptop at hand). However, for some tasks, more than one location could be used. For example, for the week shopping somebody could use several alternative malls or supermarkets and the final choice may depend on the proximate tasks. So, we foresee to enhance SELFPLANNER with classes of locations (actually, an ontology of locations) and give the user the possibility to select either a specific location or a class of locations for each task.
- *Meeting arrangement*. Currently SELFPLANNER does not support automatic meeting arrangement. We foresee to enhance SELFPLANNER with an automatic meeting arrangement function, where one user invites other users to a meeting, with the meeting time and location being not fully determined, and this initiates a fully- or semi-automatic negotiation phase between the users that results in specifying the meeting details.

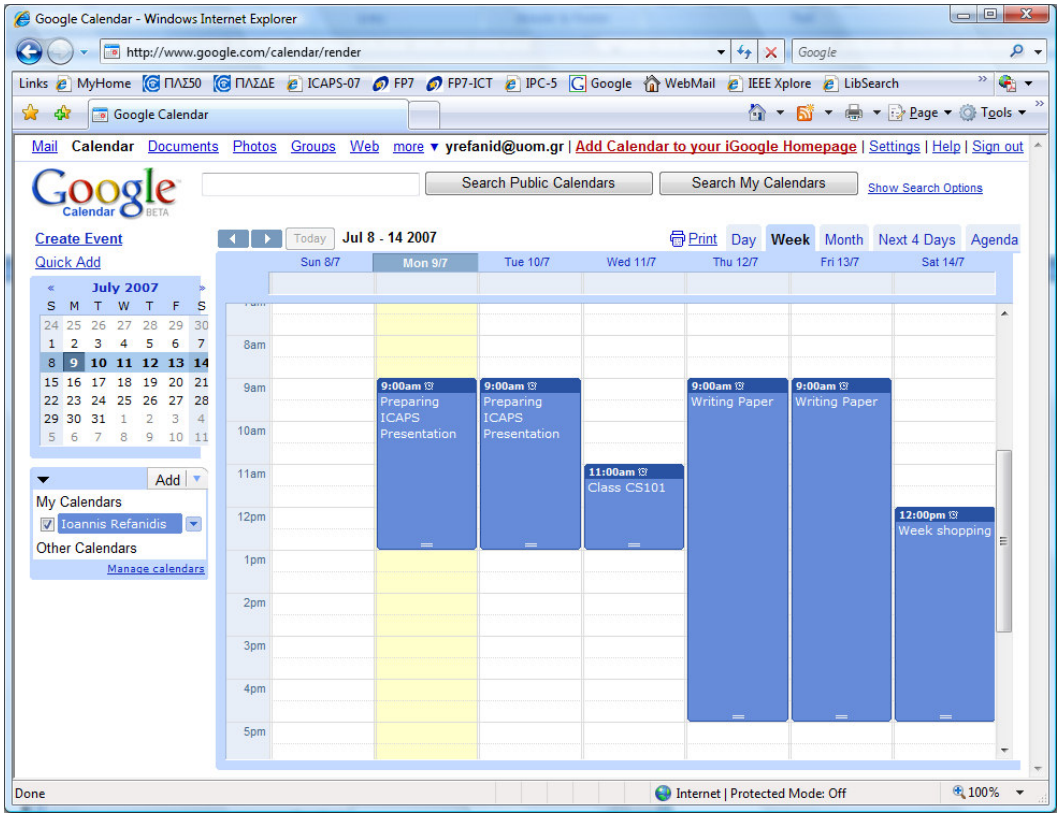

Figure 7: User's plan is displayed at Google calendar.

- *New types of constraints and preferences*: Apart from the unary constraints and preferences, currently only binary ordering constraints and binary proximity preferences are supported. New types of constraints and preferences could be added to SELFPLANNER, such as non-monotonic unary preferences, proximity constraints, ordering preferences and higher order constraints and preferences as well. Of course, for any new type of constraints and preferences, the underlying scheduling problem needs to be solved; however, the proposed SWO scheduling framework is very flexible and easily extensible and we believe that many of these new functionalities will be incorporated smoothly.
- *Resource management*: Apart for managing a user's time, SELFPLANNER could be used for managing resources. Such resources may include halls (e.g. a meeting room), equipment (e.g. a video projector) etc. A task may require specific resources, so the availability of these resources will affect the task's domain. Due to the incremental nature of this problem, resources could be reserved on a first-come-first-served basis.
- *Maintenance of the user's status*: Currently, SELFPLANNER knows only the user's location as well as her upcoming tasks. In a more elaborated version of SELFPLANNER, the system should possess more information about the user, such as whether she carries her laptop, she drives her car or travels with the bus etc.
- *Action schemas*: Maintaining a user's status gives the possibility to transform the scheduling problem into a planning one. Tasks are enhanced with preconditions and effects, thus inserting a task A with preconditions $prec(A)$ into the task list may result in the automated insertion of other tasks into the task list, in order to support the open preconditions of $prec(A)$. Of course, suitable planning algorithms should be used to solve the planning problem in this case.

References

Joslin, D.E. and Clements, D.P. "Squeaky Wheel" Optimization. Journal of Artificial Intelligence Research, vol. 10 (1999), 375-397.

Refanidis, I., McCluskey, and T.L., Dimopoulos, Y.: Planning Services for Individuals: A New Challenge for the Planning Community. Workshop on Connecting Planning Theory with Practice, Whistler, British Columbia, Canada, 2004.

Refanidis, I.: Managing Personal Tasks with Time Constraints and Preferences. In Procs. of 17th International Conference on Automated Planning and Scheduling Systems (ICAPS-2007). Providence, Rhode Island, 2007.