Generating Alternative Plans for Scheduling Personal Activities

Anastasios Alexiadis and **Ioannis Refanidis**
Department of Applied Informatics, University of Macedonia,
Egnatia 156, 54006, Thessaloniki, Greece.
talex@java.uom.gr, yrefanid@uom.gr

Abstract

Alternative plan generation can be used to meet the user's preferences in a scheduling problem, when either the scheduling model does not take every available preference into account, or the user does not specify his preferences correctly in a formal manner. In this paper, an alternative plan generation method that takes into account in-domain specific attributes of the scheduling problem, is applied to an electronic calendar management scheduler in order to generate significantly different qualitative alternative plans. The scheduler is based on an adaption of the Squeaky Wheel Optimization Framework (SWO) and addresses the problem of optimizing individual activity plans (that concern activities one person has to accomplish independently of others), based on a rich model for their specification involving complex constraints and preferences and enhanced with extra attributes that measure the distance of each plan from the already found.

Introduction

Generating alternative plans for planning and scheduling problems is another way of ensuring that at least one plan will meet the user's preferences. According to Kambhampati (Kambhampati 2007), in many real world planning scenarios the user's preferences are either unknown or at best partially specified. Other systems attempt to elicit user preferences in a non-intrusive manner, by presenting alternative plans to the user and building a preference model based on his choices (Berry et al. 2011), (Myers et al. 2007).

Intelligent assistance with time and task management has been targeted by many AI researchers (Myers et al. 2007), (Freed et al. 2008), (Refanidis 2007) (Refanidis and Alexiadis 2011), (Berry et al. 2011), (Bank et al. 2012). Electronic calendar applications are usually based on a series of fully specified and independent events. These events are specified by a fixed start-time, a duration for the event and usually a location. In addition, many systems support tasks. These represent individual commitments potentially having a deadline to be met (e.g., preparing for a lecture or planning for a trip). Tasks are kept in separate task lists and do not have a specific start time. Conversion of a task to event is, usually straightforward, accomplished by dropping the task into the electronic calendar.

(Refanidis and Yorke-Smith 2010) presents a model that treats events and tasks in a uniform way. They are specified as *activities* in the model and are characterized by a number of attributes, such as a temporal domain, a duration range, a set of alternative locations, interruptibility, utilization, preferences over the temporal domain and alternative durations, constraints and preferences over the way the parts of an interruptible activity are scheduled in time. The model also supports binary constraints (ordering, proximity and implication), as well as preferences between pairs of activities, thus specifying a Constraint Optimization Problem (COP). In the same work a scheduler, based on the Squeaky Wheel Optimization framework (SWO) and combined with domain-dependent heuristics to automatically schedule activities, is presented. (Alexiadis and Refanidis 2012) presents a post-processing module for SWO that increases solution quality via local-search post-optimization.

There are a number of ways for introducing alternative plan generation in a planner or scheduler. In the literature we found two main approaches for tackling this issue. One being the management of either the set of states or the order in which they are evaluated by the search algorithm (Roberts et al. 2012), (Dechter, Flerova, and Marinescu 2012). The other—which we based our alternative generation method on—being the modification of the heuristic of the planning algorithm (Coman and Muñoz-Avila 2011), (Nguyen et al. 2012).

In this paper we present our approach to generate qualitatively, significantly different alternative plans, based on the enhancement of the plan evaluation function used by the scheduler during the optimization process with additional attributes that measure the differences with the already generated solutions. We define the plan difference function that takes into account domain-specific traits of the problem formulation, as defined in the SWO article (Refanidis and Yorke-Smith 2010).

The rest of the paper is structured as follows. Firstly, we formulate the optimization problem and illustrate the SWO-based approach. Next, we present the new evaluation function that considers (among other criteria) differences with one generated plan. We proceed to define the plan difference function (*PDiff*) for comparing two plans. Afterward, we extend the above functions to work with an arbitrary number of pre-generated solutions. In the Evaluation Section we

demonstrate their usage on a number of randomly generated problem instances. Finally, we conclude the paper and identify directions of future work.

Background

In this Section we present the problem formulation, as well as the SWO approach to cope with the problem.

Problem Formulation

In previous work (Refanidis and Yorke-Smith 2010), time is considered a non-negative integer, with zero denoting the current time. A set T of N activities, $T = \{T_1, T_2, \ldots, T_N\}$, is given. For each activity $T_i \in T$, its minimum duration is denoted with d_i^{min} and its maximum duration with d_i^{max}. The decision variable p_i denotes the number of parts in which the *i-th* activity has been split, with $p_i \geq 1$. T_{ij} denotes the *j-th* part of the *i-th* activity, $1 \leq j \leq p_i$. The sum of the durations of all parts of an activity must be at least d_i^{min} and no greater than d_i^{max}.[1] For each T_{ij}, the decision variables t_{ij} and d_{ij} denote its start time and duration. The sum of all d_{ij}, for a given i, must equal d_i.[2] Non-interruptible activities are scheduled as one part.

For each T_i, we define the minimum and maximum part duration $smin_i$ and $smax_i$,[3] as well as the minimum and maximum temporal distances between every pair of parts, $dmin_i$[4] and $dmax_i$.[5]

For each activity T_i, its temporal domain is defined as a set of temporal intervals defining $D_i = [a_{i1}, b_{i1}] \cup [a_{i2}, b_{i2}] \cup \ldots \cup [a_{i,F_i}, b_{i,F_i}]$, where F_i is the number of intervals of D_i.[6]

A set of M locations, $Loc = \{L_1, L_2, \ldots, L_M\}$, as well as a two dimensional, not necessarily symmetric, matrix $Dist$ that holds the temporal distances between locations are given. Each activity T_i has a set of possible locations $Loc_i \subseteq Loc$, where its parts can be scheduled. The decision variable $l_{ij} \in Loc_i$[7] denotes the particular location where T_{ij} is scheduled.[8]

Activities may overlap in time. Each activity T_i is characterized by a utilization value, $utilization_i$.[9] At any exact time point, the set of scheduled activities should have compatible locations (i.e., locations with no temporal distance to each other as a person cannot be in two places at the same time) and the sum of their utilization values should not exceed the unit (the time point *maximum* utilization value).

The model supports four types of binary constraints: Ordering constraints, minimum and maximum proximity constraints and implication constraints. An ordering constraint between two activities T_i and T_j, denoted with $T_i < T_j$, implies that no part of T_j can start its execution before all parts of T_i have finished.[10] A minimum (maximum) distance binary constraint between activities T_i and T_j implies every two parts, one of T_i and another of T_j, must have a given minimum (maximum) temporal distance.[11] Finally, an implication constraint of the form $T_i \Rightarrow T_j$ implies that in order to include T_i in the plan, T_j should be included as well.[12]

Scheduling personal activities is considered a constraint optimization problem. That said, the empty schedule is a valid schedule but with low utility, thus we are interested in better schedules. There are several sources of utility. The main source concerns the activities themselves. Each activity T_i included in the schedule contributes utility $U_i(d_i)$ that depends on its allocated duration. The way T_i is scheduled by a schedule π_i within its temporal domain constitutes another source of utility, $U_i^{time}(\pi_i)$. The user can define linear and stepwise utility functions of time over the temporal domain of each activity.

Any form of hard constraint can also be considered a soft constraint that might contribute utility. So, minimum and maximum distance constraints between the parts of an interruptible activity might contribute $U_{dmin_i}(\pi_i)$ and $U_{dmax_i}(\pi_i)$ respectively. Similarly, binary preferences can be defined as well over the way pairs of activities are scheduled. Especially for ordering and proximity preferences, partial satisfaction of the preference is allowed. The Degree of Satisfaction for a partial preference p, denoted with $DoS(p)$, is defined as the ratio of the number of pairs of parts, one from T_i and another from T_j, for which the binary preference holds, to the total number of pairs of parts.

To summarize, the optimization problem is formulated as follows:

Given:

1. A set of N activities, $T = \{T_1, T_2, \ldots, T_N\}$, 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.
2. A two-dimensional matrix with temporal distances between all locations.

---

[1] $\forall T_i,\ d_i^{min} \leq d_i \leq d_i^{max}$ OR $d_i = 0$ $(C1)$

[2] $\forall T_{ij}, \sum_{j=1}^{p_i} d_{ij} = d_i$ $(C2)$

[3] $\forall T_{ij}, smin_i \leq d_{ij} \leq smax_i$ $(C3)$

[4] $\forall T_{ij}, T_{ik}\ j \neq k \Rightarrow t_{ij} + d_{ij} + dmin_i \leq t_{ik} \vee t_{ik} + d_{ik} + dmin_i \leq t_{ij}$ $(C4)$

[5] $\forall T_{ij}, T_{ik}\ j \neq k \Rightarrow t_{ij} + dmax_i \geq t_{ik} + d_{ik} \wedge t_{ik} + dmax_i \geq t_{ij} + d_{ij}$ $(C5)$

[6] $\forall T_{ij}, \exists k, 1 \leq k \leq F_i : a_{ik} \leq t_{ij} \leq b_{ik} - d_{ij}$ $(C6)$

[7] $l_{ij} \in Loc_i$ $(C7)$

[8] $\forall T_{ij}, T_{mn}, T_{ij} \neq T_{mn} \wedge (Dist(l_{ij}, l_{mn}) > 0 \vee Dist(l_{mn}, l_{ij}) > 0) \Rightarrow t_{ij} + d_{ij} + Dist(l_{ij}, l_{mn}) \leq t_{mn} \vee t_{mn} + d_{mn} + Dist(l_{mn}, l_{ij}) \leq t_{ij}$ $(C8)$

[9] $\forall t, \sum_{T_{ij}:\ t_{ij} \leq t < t_{ij} + d_{ij}} utilization \leq 1$ $(C9)$

[10] $\forall T_i, T_j, T_i < T_j \Leftrightarrow d_i > 0 \wedge d_j > 0 \Rightarrow \forall T_{ik}, T_{jl}, t_{ik} + d_{ik} \leq t_{jl}$ $(C10)$

[11] $\forall T_{ik}, T_{jl}, t_{ik} + d_{ik} + dmin_{ij} \leq t_{jl} \vee t_{jl} + d_{jl} + dmin_{ij} \leq t_{ik}$ $(C11)$

$\forall T_{ik}, T_{jl}, t_{ik} + dmax_{ij} \geq t_{jl} + d_{jl} \wedge t_{jl} + dmax_{ij} \geq t_{ik} + d_{ik}$ $(C12)$

[12] $\forall T_i, T_j, T_i \Rightarrow T_j \Leftrightarrow d_i > 0 \Rightarrow d_j > 0$ $(C13)$

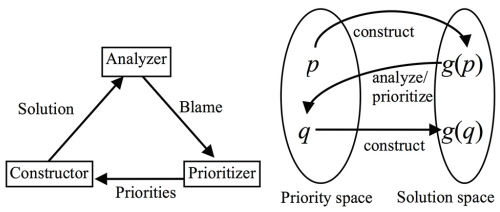

Figure 1: (a) The SWO cycle. (b) Coupled search spaces

3. A set C of binary constraints (ordering, proximity and implication) over the activities.
4. 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 d_{ij} and their locations l_{ij}, while trying to maximize the following objective function:

$$U = \sum_{\substack{i \\ d_i \geq d_i^{min}}} (U_i(d_i) + U_i^{time}(\pi_i) + U_i^{dmin}(\pi_i) + U_i^{dmax}(\pi_i)) + \sum_{p(T_i,T_j) \in P} u_p \times DoS(p(T_i, T_j)) \quad (1)$$

subject to constraints (C1) to (C13).

The SWO Approach

(Refanidis and Yorke-Smith 2010) solves the problem using the Squeaky Wheel Optimization (SWO) framework (Joslin and Clements 1999). At its core, SWO uses a Construct/Analyze/Prioritize cycle as shown in Figure 1(a). The solution is found by a greedy approach, where decisions are based on an order of the tasks determined by a priority queue. The solution is then analyzed to obtain the tasks that cannot be scheduled. Their priorities are increased, enabling the constructor to deal with them earlier on the next iteration. The cycle will be repeated till a termination condition occurs. The algorithm searches in two coupled spaces, as shown in Figure 1(b). These are the priority and solution spaces. Changes in the solution space are caused by changes in the priority space. Changes in the priority space occur as a result of analyzing the previous solution and using a different order of the tasks in the priority queue. A point in the solution space represents a possible solution to the problem.

SWO can easily be applied to new domains. The fact that it gives variation on the solution space makes it different than more traditional local search techniques such as WSAT (Selman, Kautz, and Cohen 1995). (Refanidis and Yorke-Smith 2010) presents how SWO was adapted to to the Constraint Optimization Problem presented in the previous Section.

Generating Mutiple Plans

In this Section we present our work on generating and evaluating qualitatively, significantly different alternative plans. We begin by presenting a method to generate the first alternative plan (after the standard one is found) and we continue with extending the method so as it can generate an arbitrary number of alternative plans.

Generating Qualitatively, Significantly Different Alternative Plan

Both the model and the scheduler have been designed with the assumption of generating *one* high-quality schedule for a set of activities. For generating an arbitrary number of qualitatively, significantly different alternative schedules, so as the user can choose one according to her preferences, we extended the model so as to integrate the notion of the *value* of an alternative plan.

We define the value of an alternative plan as a linear combination of its utility (1) with its deviation from the already generated plans. In the simplest case, when there is a single generated plan π' to which we want to compare an alternative plan π, we adopt the following formula for evaluating π, as a replacement to formula (1):

$$V(\pi) = U(\pi) + C \times U(\pi') \times PDiff(\pi, \pi') \quad (2)$$

where C is a constant that weights π's difference over π' and $PDiff(\pi, \pi')$ is a function assessing the two plans' differences. As it will be shown later in this Section, $PDiff(\pi, \pi')$ ranges between 0 and 1. The higher $PDiff(\pi, \pi')$ is, the greater the differences between the two plans; $PDiff(\pi, \pi') = 0$ stands for no differences.

In the initial stage, when the main plan hasn't been generated yet, $U(\pi') = 0$, so the above formula is simplified to $V(\pi) = U(\pi)$, thus resulting in no differences over the main plan generation procedure. When the scheduler has already found a plan for the user, it will try to maximize concurrently both the utility of the current plan, $U(\pi)$, as well as the deviation from the already found plan, $C \times U(\pi') \times PDiff(\pi, \pi')$. Note that the latter term of the sum depends both on the difference between the two plans and on the quality of the already found one. The factor of $U(\pi')$ is introduced in order to scale two terms of the sum similarly.

Quantifying the Degree of Deviation Between Two Plans

We measure the deviation between two plans, π and π', by function $PDiff(\pi, \pi')$, which takes into account the following metrics:

1. The change in the total duration of each plan's activity.
2. The change in the location of each plan's activity or part of it.
3. The change in the time windows where the various parts of each activity have been scheduled.
4. The change in the order in which pairs of activities have been scheduled.

In order to define precisely the above metrics, we introduce two extra functions. First we define function $\tau(\pi, T_i, x) = t \in D_i$, which for a schedule π, an activity T_i and time-slot index x, $1 \leq x \leq d_i$, maps it into the *x-th* time-slot of that activity, as scheduled in π. Time-slots are individual discrete points of time. So, this function maps the *order* of a time-slot of an activity in the solution, to the absolute time where this time slot has been scheduled. Similarly, we define function $\lambda(\pi, T_i, x) = l \in Loc_i$, which maps the triple (π, T_i, x) to the location where the *x-th* time-slot of T_i has been scheduled, according to π.

Durations Deviation: For the activities $T_i \in T$, which appear in at least one of the two plans, π and π', the total duration deviation between the two plans is computed according to formula:

$$\Delta D = \frac{\sum_i |d_i - d'_i|}{\sum_i max(d_i, d'_i)} \tag{3}$$

where d_i is the duration of T_i in plan π and d'_i is the duration of T_i in plan π'. ΔD ranges between 0 and 1. If T_i appears in one of the two plans, its duration in the plan that does not appear in is considered zero.

Locations Deviation: For the activities $T_i \in T$, which appear in at least one of the two plans, π and π', the total location deviation between the two plans is computed according to formula:

$$\Delta L = \frac{1}{\sum_i min(d_i, d'_i)} \times \sum_i \sum_{\substack{x=1 \\ \lambda_x^{T_i} \neq \lambda'^{T_i}_x}}^{min(d_i, d'_i)} 1 \tag{4}$$

where $\lambda_x^{T_i} = \lambda(\pi, T_i, x)$ and $\lambda'^{T_i}_x = \lambda(\pi', T_i, x)$. ΔL ranges between 0 and 1. If T_i appears in one of the two plans only, ΔL is set equal to 1.

Absolute Time Deviation: For each activity $T_i \in T$, which appears in both plans π and π', the total absolute time deviation for T_i among the two plans is computed according to formula:

$$\Delta Time_i = \sum_{x=1}^{min(d_i, d'_i)} \frac{|\tau_x^{T_i} - \tau'^{T_i}_x|}{max_{\tau_{D_i}} - min_x(\tau_x^{T_i}, \tau'^{T_i}_x)} \tag{5}$$

where $max_{\tau_{D_i}}$ is the is the maximum time-slot of the domain of activity T_i, $\tau_x^{T_i} = \tau(\pi, T_i, x)$ and $\tau'^{T_i}_x = \tau(\pi', T_i, x)$. $\Delta Time_i$ ranges between 0 and 1. For activities T_i appearing in one only of π and π', we define $\Delta Time_i = 1$. On the other hand, for activities T_i appearing in none of π and π', we define $\Delta Time_i = 0$.

The overall absolute time deviation is defined as:

$$\Delta Time = \sum_{i=1}^{N} \Delta Time_i \tag{6}$$

Ordering Differences: For each pair of activities, T_i and $T_j \in T$, which both appear in a plan π, the precedence of T_i over T_j in π is computed as:

$$\nu_{ij} = \frac{1}{d_i \times d_j} \sum_{x=1}^{d_i} \sum_{y=1}^{d_j} \begin{cases} 1 \text{ if } \tau_x^{T_i} \leq \tau_y^{T_j} \\ 0 \quad \text{otherwise} \end{cases} \tag{7}$$

That is, ν_{ij} represents the percentage of pairs of parts, one from T_i and one from T_j, such that the part of T_i is scheduled not later than the part from T_j. ν_{ij} ranges between 0 and 1. It T_i appears in π but T_j does not appear, we define $\nu_{ij} = 1$. If T_i does not appear in π (irrelevant to whether T_j appears in π or not), we define $\nu_{ij} = 0$.

Subsequently, the ordering deviation for a pair of activities, T_i and T_j in two plans, π and π' is defined as:

$$\Delta O_{ij} = |\nu_{ij} - \nu'_{ij}| \tag{8}$$

where ν'_{ij} refers to plan π'. ΔO_{ij} ranges between 0 and 1.

Finally, we define the total ordering deviation as:

$$\Delta O = \frac{2}{N \times (N-1)} \times \sum_{i=1}^{N} \sum_{j=i+1}^{N} |\nu_{ij} - \nu'_{ij}| \tag{9}$$

ΔO also ranges between 0 and 1.

Plan Difference Between Two Plans

Based on the above formulas, we define the plan difference between two plans, $PDiff(\pi, \pi')$ as:

$$PDiff(\pi, \pi') = \begin{array}{c} \Delta D \times W_D + \Delta L \times W_L \\ + \Delta Time \times W_{Time} + \Delta O \times W_O \end{array} \tag{10}$$

where W_D, W_L, W_{Time} and W_O are non-negative weights and $W_D + W_L + W_{Time} + W_O = 1$. The user can specify the values of the weights, thus specifying his preferences on multiple plan generation (thas is, emphasizing his priorities over the above attributes of an already found plan). If an activity is not scheduled in one of the two plans, it obtains the full difference penalty as it is considered to deviate in all the above methods.

Generating More than Two Alternative Plans

In order to generate more than two plans, we extended formula (2) as follows:

$$V(\pi) = U(\pi) + C \times \frac{\sum_{\forall \pi' \in \Delta} U_{\pi'} \times PDiff(\pi, \pi')}{|\Delta|} \tag{11}$$

where Δ is the set of the already generated plans and C is a parameter representing the weight standing for the user's preference on the deviation from the already found plans versus the utility of the alternative plan.

The new formula for calculating the utility $V(\pi)$ (formula 9) is simplified back to formula (2), when comparing between only two plans (the one being evaluated and one pregenerated). When generating the original plan it is simplified back to the original utility function (1).

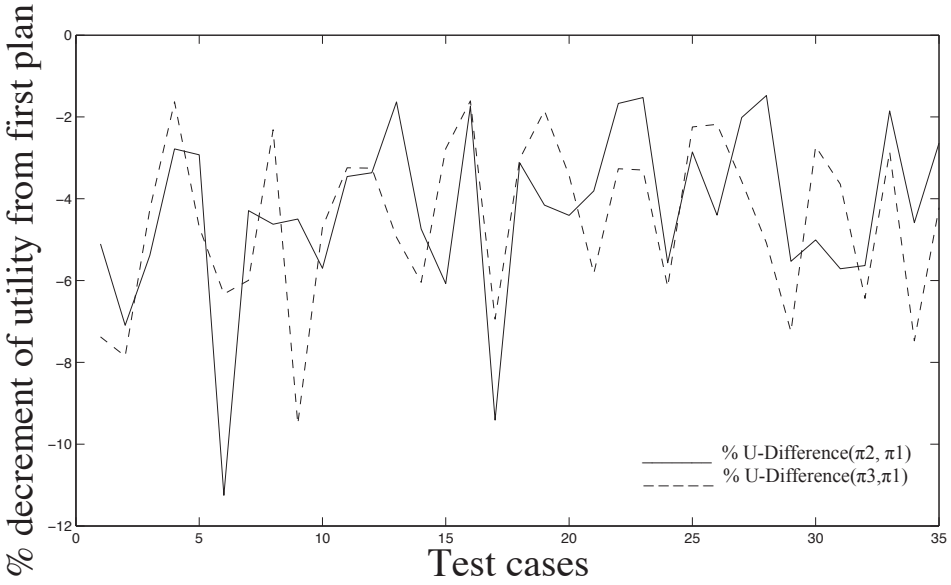

Figure 2: Percentage difference of utility between the alternative plans and the original

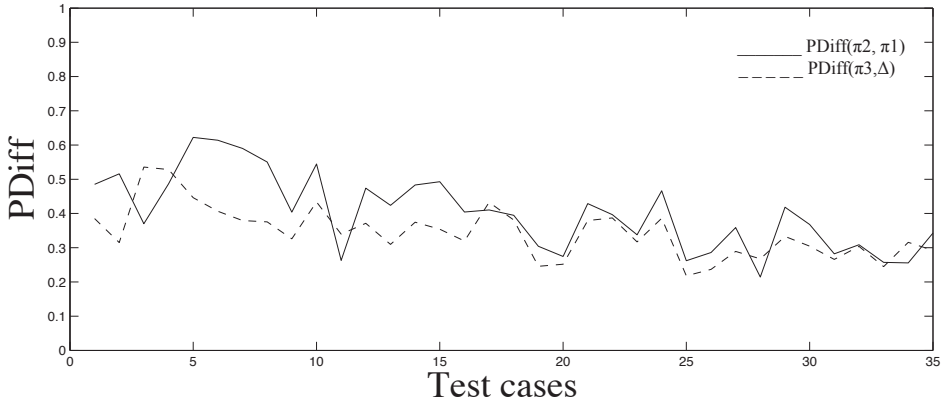

Figure 3: PDiff values for the two alternative plans for each test case

Evaluation

The scheduler SWO (written in C++) was extended to include the alternative evaluation function V (using $PDiff$), when generating plans and to allow the generation of more than one plan, by means of restarting itself. At each restart a dynamic list Δ, holding the already generated plans, gets updated.

We generated three plans, on 35 random test cases, ranging in size from problem instances of six activities to thirty six in steps of five. We set the parameters used for the alternative plan generation as follows: $C = 1.0, W_D = W_L = W_{Time} = W_O = 0.25$.

In Figure 2 we show the utility differences between the three plans, for each of the 35 problem instances of the test set. The solid line represents the utility percentage difference between the second plan (first alternative) and the original plan, whereas the dashed line represents the utility percentage difference between the third plan and the original. The original's (first plan's) quality is represented by the top line. We use the original utility function (formula 1) when comparing plan quality, as shown in the figure. V is used for the generation phase only, when an intermediate solution gets evaluated while the scheduler optimizes the plan. The two alternative plans are of a slight lower utility value than the original plan though they are not far below in quality. The drop in utility for the second plan, in comparison to the first, ranges from 1.47% to 11.25%. For the third plan the minimum and maximum drops are 1.6% and 9.51% respectively. The average drop for the second plan is 4.2% and of the third 4.5%.

In Figure 3 we represent the *PDiff* values for the two alternative plans, the first alternative comparing to the origi-

nal plan, and the second alternative comparing to the original plus the first alternative. The difference of each alternative plan to the original depends heavily on each problem instance's specific attributes, such as the available domain for each activity (more so than the number of activities in the problem instance), how strong (in utility values) are the preferences set on it and how many of them exist, as well how they intertwine. The minimum *PDiff* value was one of 0.2179, in the second (after the original) alternative plan of a problem instance. The maximum value was one of 0.6224, in the first (after the original) alternative plan of another problem instance.

One looking at the actual alternative solutions immediately sees apparent changes in many of the activities' parts temporal positions (the most common change), as well as their durations (another common change when applicable) and locations (rarest, as location selection don't provide a utility to standard SWO, except in the alternative plan generation phase). Changing the C value parameters will shift the preference to either more different plans (of lower standard utility) or similar ones to the original.

Conclusions and Future Work

The SWO scheduler with the alternative plan generation method, presented in this paper, manages to generate an arbitrary number of alternative plans for a user's scheduling problems, so the user can choose the best one suited to her needs. It is based on the premise that a user will not always be able to specify her constraints and preferences correctly in the formal model—which has been found to be the case in many real situations. In such cases, offering a number of alternative plans to the user provides her with more possibilities for picking a plan according to her actual preferences. A number of parameters for this method are also customizable by the user, enabling her to choose which in-domain characteristics she considers more important.

This paper presents our work on an alternative plan generation method, which can be potentially enriched with the exploration of more characteristics of domain-specific attributes of the scheduling problem being solved. Particularly, location differences should be further examined, as the model does not support location preferences but only alternative location options. Other domain dependent attributes to measure the deviation between two plans, either at the activity level or at a global level, could be considered as well. Last of all, alternative plans of user-given problem instances will be evaluated.

Acknowledgements

The authors are supported for this work by the European Union and the Greek Ministry of Education, Lifelong Learning and Religions, under the program "Competitiveness and Enterprising" for the areas of Macedonia-Thrace, action "Cooperation 2009", project "A Personalized System to Plan Cultural Paths (myVisitPlanner^{GR})", code: 09SYN-62-1129.

References

Alexiadis, A., and Refanidis, I. 2012. Meeting the objectives of personal activity scheduling through post-optimization. First International Workshop on Search Strategies and Non-standard Objectives (SSNOWorkshop'12), in conjunction with CPAIOR-2012, Nantes, France.

Bank, J.; Cain, Z.; Shoham, Y.; Suen, C.; and Ariely, D. 2012. Turning personal calendars into scheduling assistants. In *Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems Extended Abstracts*, CHI EA '12, 2667–2672. New York, NY, USA: ACM.

Berry, P. M.; Gervasio, M.; Peintner, B.; and Yorke-Smith, N. 2011. Ptime: Personalized assistance for calendaring. *ACM Trans. Intell. Syst. Technol.* 2(4):40:1–40:22.

Coman, A., and Muñoz-Avila, H. 2011. Generating diverse plans using quantitative and qualitative plan distance metrics. In Burgard, W., and Roth, D., eds., *AAAI*. AAAI Press.

Dechter, R.; Flerova, N.; and Marinescu, R. 2012. Search algorithms for m best solutions for graphical models.

Freed, M.; Carbonell, J.; Gordon, G.; Hayes, J.; Myers, B.; Siewiorek, D.; Smith, S.; Steinfeld, A.; and Tomasic, A. 2008. Radar: a personal assistant that learns to reduce email overload. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 3*, AAAI'08, 1287–1293. AAAI Press.

Joslin, D., and Clements, D. P. 1999. Squeaky wheel optimization. *J. Artif. Intell. Res. (JAIR)* 10:353–373.

Kambhampati, S. 2007. Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain models.

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.

Nguyen, T. A.; Do, M. B.; Gerevini, A.; Serina, I.; Srivastava, B.; and Kambhampati, S. 2012. Generating diverse plans to handle unknown and partially known user preferences. *Artif. Intell.* 190:1–31.

Refanidis, I., and Alexiadis, A. 2011. Deployment and evaluation of selfplanner, an automated individual task management system. *Computational Intelligence* 27(1):41–59.

Refanidis, I., and Yorke-Smith, N. 2010. A constraint-based approach to scheduling an individual's activities. *ACM TIST* 1(2):12.

Refanidis, I. 2007. Managing personal tasks with time constraints and preferences. In Boddy, M. S.; Fox, M.; and Thiébaux, S., eds., *ICAPS*, 272–279. AAAI.

Roberts, M.; Howe, A.; Ray, I.; and Urbanska, M. 2012. Using planning for a personalized security agent.

Selman, B.; Kautz, H.; and Cohen, B. 1995. Local search strategies for satisfiability testing. In *DIMACS SERIES IN DISCRETE MATHEMATICS AND THEORETICAL COMPUTER SCIENCE*, 521–532.