Scheduling and Planning algorithms for Electronic Calendar Management

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

As part of my doctorate thesis I am looking into scheduling and planning problems and systems with a focus on electronic calendar applications.

Cooperating with my supervisor Associate Professor Ioannis Refanidis, I have co-developed such a system, entitled SELFPLANNER (Refanidis and Alexiadis 2007) (Refanidis and Alexiadis 2008), which can be accessed at *http://selfplanner.uom.gr*. The system SELFPLANNER started as my master thesis. It is a scheduling system that supports simple, interruptible and periodic activities, where the user can define their minimum and maximum durations, their temporal domain, alternative locations, ordering constraints and temporal preferences. Specifically, in regards to to the temporal domain, an original method of defining an activity's domain was developed (Alexiadis and Refanidis 2009). This method combines the definition and application of templates with *manual* editing of the calendar, by the user, to specify the temporal domain.

An extended version of the scheduling problem definition was published (Refanidis and Yorke-Smith 2010) and the system was extended to incorporate that model. In the extended model, the system solves a scheduling problem, where it attempts to maximize a global utility function for the particular problem instance being solved, while meeting all the constraints of the problem. An activity can be omitted from the resulting schedule without a violation of the problem's constraints. However, doing so will result in that activity's utility contribution not being added to the resulting solution's utility value. Moreover, any binary preferences involving said activity will be ignored and thus provide no utility to the solution. The extended system also supports activities overlapping in time, based on the activities' *utilization* value.

The work resulting to the system's extension was published in the journal *Computation Intelligence* (Refanidis and Alexiadis 2011).

Alternative algorithms were designed and developed to solve SELFPLANNER's scheduling problem. The first of them was based on a *genetic algorithm* approach. Each *chromosome* of the *population* corresponds to the set of activities that define a solution for a particular problem instance. A stochastic local search algorithm, that operates on a tree structure, was developed to schedule an initial version of the solution, as well as be called upon when needed to recompute a part of the solution. That algorithm supports *backtracking*. The *fitness* function calculates the utility value of a chromosome (solution). For the *crossover* function, we combined half the activities of a chromosome with half of another one. Lastly, for the *mutation* function we delete some parts of activities from the solution and let the tree-based local search algorithm reschedule them.

Unfortunately the above algorithm gave us worse solutions than expected. Our second approach was to develop a *complete search* algorithm for the above problem. We developed a complete-search algorithm, using brunch and bound in *ECliPSe Prolog*. We defined the problem as a *Constraint Optimization Problem* (COP). The resulting system could not solve anything bigger than *toy problem instances*, though it could find the best possible solutions in simple enough problem instances.

Our next step was to generate a random solution and search the *conflict sets* of the problem's variables, attempting to optimize the search algorithm so as to not visit some variable assignments again, the ones that did not produce solutions greater of some specific utility value. That algorithm searched the space of all possible variables assignments (including constraint-violating ones), though it penalized greatly any constraint violation, with a different penalty depending the constraint's type. This approach did not produce a complete search algorithm, like the previous attempt, though it produced fairly acceptable solutions that were of a little lower quality than the ones solved by SELFPLANNER's scheduler.

Our final attempt to improve the scheduler's solution quality was through post-optimization. Post-optimization modules developed for planners recently, that exploit a plan's structure, have proven to be effective (Chrpa, McCluskey, and Osborne 2012). ARAS system (Nakhost and Müller 2010) is such recent module, where local-search is used to improve and existing valid plan, by removing unnecessary actions and by finding shortcuts—thus taking advantage of the problem's structure.

In our approach, SELFPLANNER's scheduler was used and a post-optimization module was designed. A set of

transformations operating on the decision variables of the COP problem, and shifting them in ways that took advantage of the schedule's structure, were defined and used a by a *hill-climbing* local search algorithm. It managed to give a 3.5% average improvement in solution quality on a large number (60) of difficult problem instances, with a best case of 9.8%. The results were published in a workshop (Alexiadis and Refanidis 2012).

A new transformation, that attempts to insert into the solution (being post-optimized) activities that were omitted, was added to the post-optimizing module. *Hill-climbing* search was replaced by a local stochastic search algorithm based on *Simulated Annealing*. The resulting module was faster than the previous one based on *hill-climbing* and produced better solutions. It managed to produce a 6.7% average improvement in the same set of difficult problems, while having a best case of 22.7%.

Finally, the original algorithm was extended to produce multiple solutions. A new total utility function, dependent on the old one, is being maximized by the scheduler and post-processor. The new function values and rewards differences with the solutions already produced by the scheduler.

Future Work

The next step of my research will be to focus on multi-user activity scheduling. The problem model defined in (Refanidis and Yorke-Smith 2010) will be extended to define multi-user activities. We define a *multi-user activity* as any activity that is present in more than one problem instance, where its successful scheduling depends on it being scheduled in all the problem instances that it depends on and with the same assignments to its decision variables between all the the different problem instances.

The most typical example of multi-user activities are *meetings*. Meeting scheduling literature will be reviewed and used as the starting point for a multi-user activity negotiation protocol.

Usually, such protocols depend on a number of negotiation steps between two or more parties. A common temporal domain must be decided between the parties, as well as a meeting duration and location. User privacy is a very important consideration. A user's calendar is considered a private affair and only the *minimum* required information about it, needed so as to decide on the meeting's parameters, should be transmitted.

Another issue that must be considered is the multi-user activity's *initiator*. Whoever initiates a multi-user activity (such as a meeting), will probably act in a coordinator role. While taking this into account, the protocol should still provide a degree of *fairness* to all parties involved into which time interval is chosen for the shared activity.

Conflicting preferences on the time intervals (and the location of the multi-user activity) are to be considered by the protocol. Voting methods can be used here to decide on a common interval, as they have been used for this purpose before (Antila, Mäntyjärvi, and Könönen 2010).

Previous possible models for meeting scheduling, such as the one in (Sen et al. 1997), usually take a four-step approach to this problem:

1. The host (meeting initiator) attempts to find some time intervals that suit her. She sends these intervals to the invitees.
2. Each invitee receive these intervals and tries to find local solutions that satisfy the constraints of the meeting. He then sends their proposals (as bids) back to the host. The proposals can be subsets of those sent by the host, or they can be counter-proposals.
3. The host collects the replies from the invitees and evaluates their bids. If they all (including the host) include a common interval, the meeting is scheduled for that interval and awards are sent back to the invitees. If no common interval is found, the host generates a new proposal, consisting of a set of time intervals, based on the bids of the invitees and her own calendar. She sends that proposal back to them.
4. Upon receiving the reply, the invitees reply as in step (2). If an award is received, for their bids, they mark the available time slot in their calendars. Otherwise they reject the proposed slots.

The above algorithm is repeated until a meeting is scheduled between the parties or until it is decided that a common time slot cannot be agreed upon by all parties.

In our model, the scheduler will act as a software agent for its user, representing their interests and solving its respective user's problem instance locally, attempting to maximize her total utility value.

Based on SELFPLANNER's model, the integration of interruptible multi-user activities, by the protocol, will also be investigated. This will demand a more complex protocol but will offer a more flexible model than classic meeting scheduling.

The degree of automatization by the software agent, that acts on behalf of a user, will also be considered. Machine learning techniques may be added, to provide additional automation as the software agent learns to cope with its users preferences.

My next goal, after multi-user activity scheduling, will be to extend SELFPLANNER's COP scheduling problem into a full planning problem. This will allow activities, which are not mere "black boxes" that take space in the user's calendar, but full entities with preconditions and effects, allowing a more complex and enriched model than the current one.

In our extended model, the initial state of the planning problem will be the user's current state, which will be known by the planner. This state will include the results of all his previous actions. An ontology of actions can be designed, where the user can choose from (or be suggested by the system), according to the user's current goals. The construction of plans, in this case can be done either automatically, or in a manner of *mixed initiative planning* (by communicating with the user).

Hierarchies of actions is also possible in this model, and will be examined. Different types of activities could be mapped into different classes of actions. As an example, we have activities with very low *utilization*, such as "check e-mail", that can be completed in any available location. These

could constitute one class of actions, that are easy to plan. Opposite that, we could have a class "write a document", where an instance of that class such as "write a paper for ICAPS", will be interruptible and have great demands on a user's calendar and a high *utilization* value. Moreover, it could only be completed in one or two available locations (usually office and home).

A classic high-performance planner, such as LAMA (Richter and Westphal 2010), can be used for solving the core planning problem. A post-optimization module, such as ARAS, can be applied on top as well; seeing the advantages it has provided to classical planning, but with domain-dependent heuristics instead of domain-independent ones. Our previous work on *shifting transformations* can be applied here, but enriched for the new problem model.

References

Alexiadis, A., and Refanidis, I. 2009. Defining a task's temporal domain for intelligent calendar applications. In *AIAI*, 399–406.

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.

Antila, V.; Mäntyjärvi, J.; and Könönen, V. 2010. Optimizing meeting scheduling in collaborative mobile systems through distributed voting. In *PerCom Workshops*, 238–243. IEEE.

Chrpa, L.; McCluskey, T. L.; and Osborne, H. 2012. Optimizing plans through analysis of action dependencies and independencies. In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *ICAPS*. AAAI.

Nakhost, H., and Müller, M. 2010. Action elimination and plan neighborhood graph search: Two algorithms for plan improvement. In Brafman, R. I.; Geffner, H.; Hoffmann, J.; and Kautz, H. A., eds., *ICAPS*, 121–128. AAAI.

Refanidis, I., and Alexiadis, A. 2007. Selfplanner: A intelligent web-based calendar application. Demonstrated at the demo session of the 17th International Conference on Automated Planning and Scheduling Systems (ICAPS-07). Providence, Rhode Island, US.

Refanidis, I., and Alexiadis, A. 2008. Selfplanner: Planning your time! ICAPS 2008 Workshop on Scheduling and Planning Applications. Sydney.

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.

Richter, S., and Westphal, M. 2010. The lama planner: guiding cost-based anytime planning with landmarks. *J. Artif. Int. Res.* 39(1):127–177.

Sen, S.; Sen, I.; Haynes, T.; and Arora, N. 1997. Satisfying user preferences while negotiating meetings. *Intl. Journal on Human-Computer Studies* 47:47–3.