

A Personalized Time Management Assistant: Research Directions

Pauline M. Berry^{*}, Melinda T. Gervasio^{*}, Tomás E. Uribe^{*},
Martha E. Pollack[‡] and Michael E. Moffitt[‡]

^{*}Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025
[\[berry.gervasio,uribe\]@ai.sri.com](mailto:[berry.gervasio,uribe]@ai.sri.com)

[‡]Computer Science and Engineering
University of Michigan
3401 Beal Ave.,
Ann Arbor, Michigan 48109
[\[pollackm,mmoffitt\]@eecs.umich.edu](mailto:[pollackm,mmoffitt]@eecs.umich.edu)

Abstract

This paper presents ongoing work to build the Personalized Time Manager (PTIME) system, a persistent assistant that builds on our previous work on a personalized calendar agent (PCalM) (Berry et al. 2004). PCalM was an early test of the hypothesis that in order to persist and be useful, an intelligent agent must learn and adapt to the user's changing needs. PTIME extends this idea to include more general time management, soft constraint satisfaction, richer learning, persistence, and steps towards adjustable autonomy.

Introduction

Despite a proliferation of calendar tools to organize, display, and track commitments, most people still spend a considerable amount of time personally organizing meetings and managing the constant changes and adjustments that must be made to their schedules. Automated meeting scheduling assistants have shown promise but their use tends to be fleeting, as they do not evolve over time. In addition, people use a variety of tools, such as *to-do* lists, to keep track of workload and deadlines not supported in the typical calendar tools. Our goal is to provide the technology necessary to manage an individual's temporal commitments in a consistent, integrated framework over an extended period of time.

The PTIME project is part of an ambitious automated assistant called CALO. In this context, the primary functional goal of PTIME is to provide time-management support that is personalized to the needs of individual users, and adaptive to their changing circumstances. To achieve this goal, we are designing PTIME so that

1. PTIME will unobtrusively learn user preferences through a combination of passive learning, active learning, and advice-taking;
2. As a result of (1), over time the user will become more confident of PTIME's ability, and will thus let it make more decisions autonomously; and
3. As autonomy increases, PTIME will learn when to involve the user in its decisions.

The human time management problem is characterized by its intensely personal nature. Many people—especially

busy workers—are fiercely protective of their time and reluctant to relinquish control over its management. Moreover, people have different needs and priorities regarding the reminders they receive, and different preferences and practices regarding how they schedule their time, how they negotiate with others about meetings and appointments, and the amount of information they are willing to share during such negotiation. PTIME is being developed to tune itself to these differences.

CALO exists in an open, unbounded environment where issues of privacy, authority, cross-organizational scheduling, and availability of participants abound. In (Franzin et al. 2002), complete privacy is assumed and the availability and preferences of other users are learned across time. Systems such as RCAL (Payne et al. 2002), DCOP (Pragnesh et al. 2003), and (Pragnesh and Veloso 2004) assume more cooperative environments, and use distributed constraint-solving to schedule meetings.

These systems only schedule feasible options given fixed, hard constraints of the request. In contrast, PTIME considers the process of finding the best solution as a dialogue between user and agent, and treats the underlying scheduling problem as a soft CSP. PTIME also addresses the problems of individual preference and scheduling events within the context of the user's workload and deadlines.

Figure 1 is a screenshot of the current PTIME interface, and illustrates the collaborative nature of the dialogue between PTIME and the user.

Architecture

The PTIME architecture, illustrated in Figure 2, includes a number of components that make it personalized and adaptive. Key features of the architecture include:

- A **Process Framework (PTIME-Controller)**, which captures possible interactions with users and other agents, in the form of structured decision points
- **Preference Learning (PLIANT)**, which lets the system evolve over time by learning process preferences, scheduling preferences, and, eventually, new processes from the user.

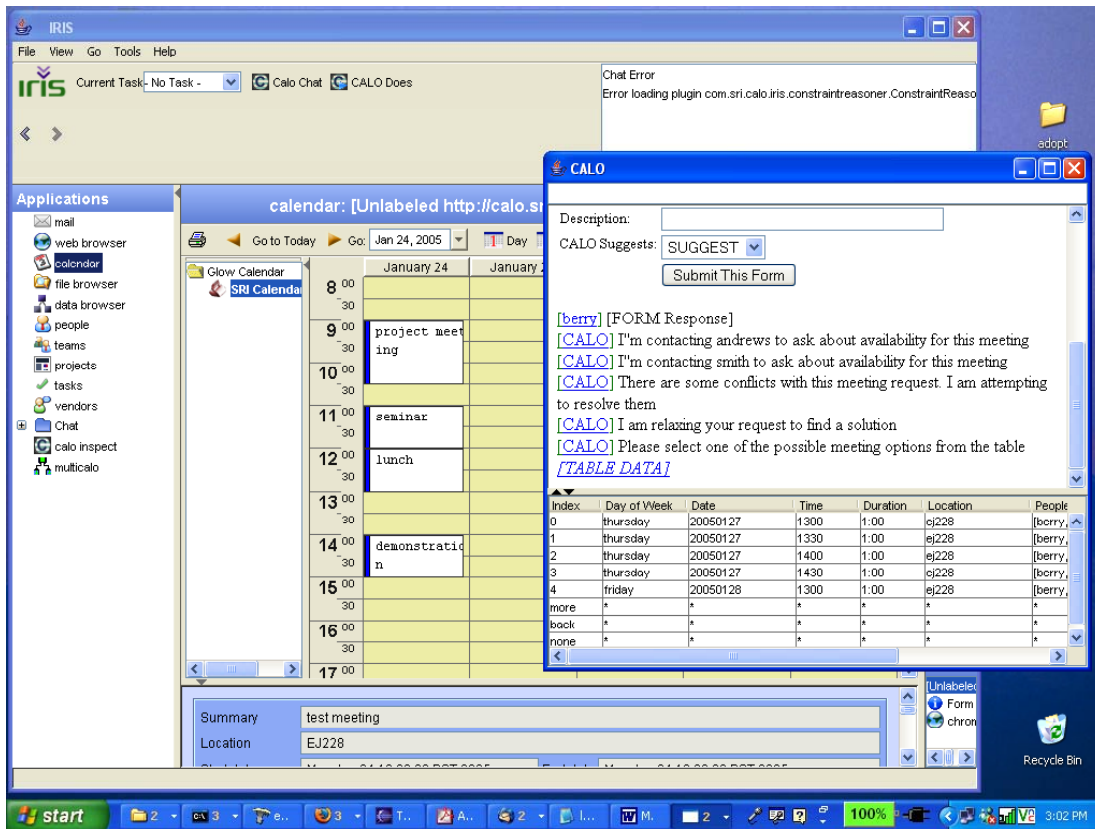


Figure 1: A screenshot from PTIME.

- **Advisability**, which enables direct instruction by the user at various levels of abstraction. Exploiting the explicit decision points in the process framework lets the user make choices and give advice. Choices may involve selecting an alternative scheduling process, e.g. *negotiate a new time for the meeting* vs. *relax an existing constraint to accept the current time*; or they may involve expressing simple temporal preferences, e.g. *don't schedule meetings just before lunch*.
- **Constraint Reasoning (PTIME-CR)**, which permits reasoning within a unified plan representation. The representation used by PTIME unifies temporal and non-temporal constraints, as well as soft and hard constraints, within a framework of preferences that allows the constraint reasoner to consider workload issues and task deadlines when scheduling typical calendar events such as meetings. The constraint reasoner uses a hybrid solver that manages the application of temporal CSP algorithms, e.g., to handle Simple Temporal Problems (STPs) (Dechter et al. 1991) and Disjunctive Temporal Problems (DTPs) (Stergiou and Koubarakis 1998, Tsamardinou and Pollack 2003), to address the complex constraint space and preference handling and to enable partial constraint satisfaction. It also provides the ability to explore alternative conflict resolution options via relaxation, negotiation, and explanation techniques.
- **Personalized Reminder Generation (PTIME-RG)**, which reasons intelligently about if, when, and how to alert the user of upcoming events or possible conflicts amongst events. This work builds on the Autominder system (Pollack et al. 2003) and the PLIANT algorithms to create reminders that are context-sensitive and personalized.
- **Adjustable Autonomy**, which modulates control over decision points as the user's preferences and normal practices are learned, and trust between the user and the system is established. The goal is to decrease the system's reliance on user interaction over time.

Persistence and Learning

Central to persistence are the application of learning technology and a framework for advisability. Through continual active learning and advice taking, PTIME constructs a dynamic preference profile containing two types of guidance:

Preferences: over *schedules* (when to reserve time and with whom), *relaxations* (which constraints, or constraint sets, are more readily relaxed) and *reminders* (when, how and about which events the user should be alerted).

Process selection and application: over existing process descriptions (e.g., negotiate or relax) and learned processes.

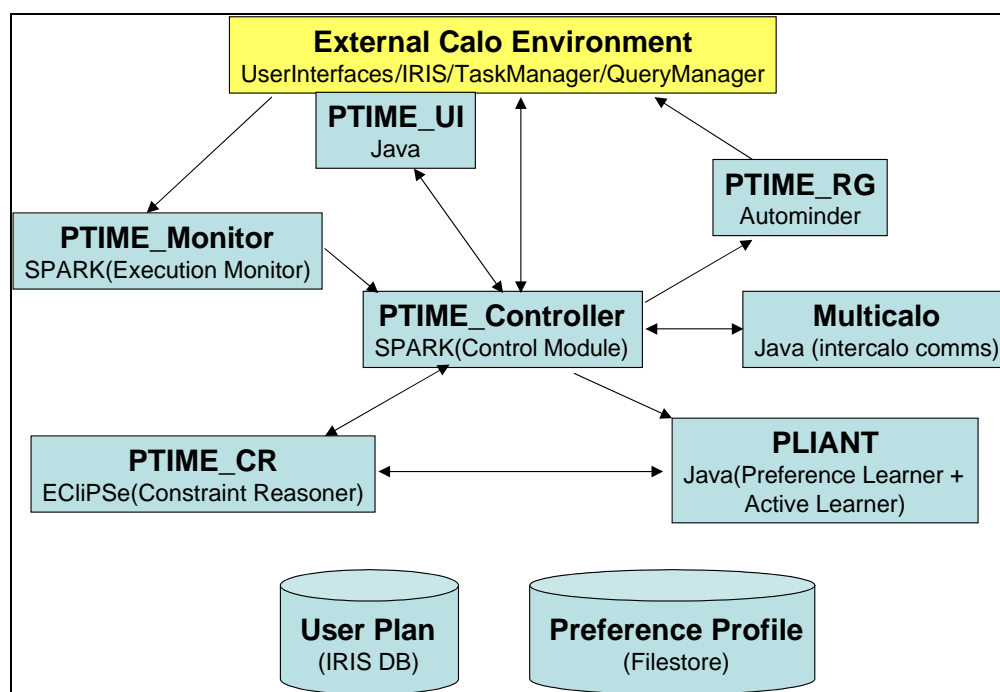


Figure 2: PTIME functional architecture

Both types of information can be actively asserted using a policy specification language, building on work on advisability and adjustable autonomy (Myers and Morley 2003). They can also be passively learned by monitoring the user's decisions.

PTIME uses a suite of tools to learn various kinds of preferences. A Support Vector Machine (SVM) module, supplemented with active learning strategies, learns user preferences about schedules in the form of an evaluation function over temporal schedule features (e.g., day of week, start time, fragmentation) (Gervasio et al. 2005). We are adding features that capture whether or not constraints are satisfied by a candidate schedule; this will let PTIME learn preferences over relaxations in the case of over-constrained schedules as well. We are also exploring procedural learning, where the performance task is to determine what to do under a particular situation rather than to evaluate the goodness of a candidate schedule. Along similar lines, we are using procedural learning to handle situations that arise after an event is scheduled: for example, if the host cannot make it or if the scheduled venue suddenly becomes unavailable. Finally, PTIME uses reinforcement learning schemes to learn both reminder strategies that are tailored to individual users and strategies for determining the amount of autonomy to take in different situations. By observing the effects of different reminder strategies on a user, PTIME can adjust its reminder strategy to account for personal traits as well as

different schedule situations. A similar process occurs with the learning of adjustable autonomy decisions.

In all cases, PTIME learns online (or from the execution traces of the user's actual interactions with PTIME), so it can continually adjust to changing user preferences and situations. Concept shift—the phenomenon of users exhibiting drastic changes in preferences—is a known issue in the calendar scheduling domain. We plan to address this problem more directly by designing a learning approach that is sensitive to sharp changes as well as a period of stabilization of user preferences over time.

Research Directions

PTIME has demonstrated its initial calendar management software within the CALO project, and is currently undergoing a test phase, conducted by an external agency, to assess its capability to learn user preferences and therefore retain a high level of usefulness to the user. PTIME development has four principal research goals for 2005, and all relate to its ability to adapt to the user's needs. In this section we describe our plans for research into hybrid constraint satisfaction and partial constraint satisfaction with preferences, our goal to design a framework and processes for negotiation between the agents and with the user, and our ongoing work to learn adjustable autonomy policies for schedule conflict situations.

Soft CSP design

The Temporal Constraint Network (TCSP) (Dechter et al. 1991) has been widely used to solve scheduling problems. Here variables represent time points, such as the start and end of events, and constraints define the range of durations and times over which these events may be scheduled. A solution is an assignment of values to variables such that all constraints are satisfied. When there is just one interval per constraint the problem is called a simple temporal problem (STP). The application of preferences STPs and the learning of those preference functions are explored by (Rossi et al. 2001). However, as indicated, a critical aspect of PTIME is our consideration of temporally disjunctive constraints. This is unique among many calendar tools and a key feature of PTIME: its ability to reason about the relationship between events on the calendar and events in the user's todo list.

A task on a user's *todo* list might be to write a project status report by a specified date. This type of task is not specifically scheduled onto a calendar, but it does imply the need for one or more spells of uninterrupted time before the due date, which should constrain the time available for scheduled meetings. This is further complicated by disjunctive relationships between events. For example, the user may wish to schedule some time to exercise, and stipulate at least half an hour between the end of exercising and the beginning of any meeting, for cooling off.

To handle these "floating" events and dependencies, the binary restrictions of STPs must be removed, resulting in a DTP. Experiments adding preferences to DTPs (Peintner and Pollack 2004) form the basis for our work on soft constraints.

However, the PTIME domain also includes non-temporal finite-domain constraints that are not efficiently handled by the DTP. For example, the user may stipulate the need for a meeting to include a set of participants defined as *at least two engineers and one account manager*, and a location that depends on the final number of participants. To further complicate matters (for the system, but not for the user), we consider the constraints to be soft, with preference functions over their relaxation. Thus, we need to combine the DTPP and STPP representations and algorithms into one constraint engine.

To solve the soft CSPs for mixed domains, we propose to combine existing solvers for the temporal and nontemporal subproblems in a hybrid formulation. Constraint solving in PTIME is implemented in ECLiPSe (Cheadle et al. 2003), a system well suited to hybrid solving. We will extend some initial experiments to explore the use of DTPPs for each case, partitioning the variables between temporal and nontemporal ones. We will also consider local search techniques to produce qualitatively different solutions to present to the user.

CSP Control

The constraint problem in PTIME is a combination of three factors: the user's existing schedule, the meeting request, and the interactive collaboration between PTIME and the user. The user may interact with PTIME to explore possible relaxed solutions to the problem, leading to a sequence of related soft Constraint Optimization Problems (COPs) to solve. For example, the user may initially specify a strong preference against meetings on Monday mornings. Later, she may weaken this preference but increase the importance of the specified meeting room.

We will model this situation by mapping the user's preferences into both the shape and height of the semiring preference functions on the relevant constraints. The shape models how much and in what way the constraint may be relaxed, and the height models the importance of the constraint. This builds on the work by (Peintner and Pollack 2004) and (Bistarelli, Montanari, and Rossi 2001).

Negotiation: Process Design for Conflict Resolution

The work on extending the constraint representation and relaxation framework of our CSP is to enable more informative dialogue between the human user and the agent. The motivation behind PTIME is to facilitate a collaborative assistant for time management. Taking note of research in collaboration (Grosz and Kraus 1999) and collaborative interfaces (Babaian, Grosz, and Shieber 2002) we will view conflict resolution as a joint task to be undertaken between the human and his agent, or between agents. Currently, the interaction is explicitly captured in the highly reactive process descriptions offered by SPARK-L (Morley 2004) and applied within a framework of advice. We would like to abstract and possibly learn the applicability conditions within the context of the dialogue.

Figure 3 presents a typical dialogue that might take place between a user and PTIME. To enable this type of dialogue, the processes capture the key decision points. Future research will construct a collaborative framework within which these processes will operate. Figure 4 illustrates an example process in SPARK-L.

At each decision point there is the possibility to automate the decision, ask the user for advice or decision, postpone the decision, or take another action. For example, when `[do: (select_solution $resultset $result)]` is the goal, a set of different actions might be intended, including asking the user to select an option or automatically selecting the highest valued option. The choice of action depends on the user's preference (learned or told), the physical context (such as the user's current activity), and the cognitive context. Learning how and when to apply each activity is a highly personalized and evolving problem. Thus, the final research direction discussed in this paper is adjustable autonomy.

User Helen: "Please schedule a group meeting early next week"

PTIME Agent: "Your specific request conflicts with your current workload and meeting constraints"

PTIME Agent: "May I suggest some possible alternatives"

1. Meet Monday at 10am without "Bob"
2. Meet Tuesday at 4pm overlapping the seminar
3. Meet Monday at 10am warning your report deadline may be in jeopardy
4. Meet Tuesday at 11 and reschedule your meeting with the boss

User Helen: I don't mind overlapping some meetings – show me more possibilities like 2.

PTIME Agent: "Ok How about"

1. Meet Monday at 11:30 running into lunch by 15 minutes
2. Meet Tuesday at 9:30 but Bob may have to leave early

User Helen: "Ok go ahead with 2"

Figure 3. Example user-agent dialogue

```
{defprocedure "schedule"
  cue: [do: (schedule $event_type $constraints $attributes)]
  preconditions (Event_Type "meeting")
  body: [context (and (User $self)
    (Participants $constraints $pset))
    seq:
    [do: (retrieve_availability $pset $constraints)]
    [do: (solve_schedule $constraints $resultset)]
    [do: (select_solution $resultset $result)]
    [select: (= $result {})]
    [do: (resolve_conflict $constraints $result)]
    [do: (confirm_meeting $result $attributes)]]
}
```

Figure 4. Example SPARK-L process

Learning for Adjustable Autonomy

As noted earlier, a central goal for PTIME is to become increasingly autonomous as its user gains trust in its ability to manage his or her time. However, it is unlikely that a user will ever want to let PTIME make all decisions autonomously, and thus one of our goals is to design a system that is adjustably autonomous. For example, when PTIME receives a request to schedule a meeting, it must decide to do one of the following:

1. Reject the request outright.
2. Accept the request and schedule the meeting, ignoring (for the time being) any conflicts this may introduce into the schedule.

3. Accept the request and adjust the schedule to remove any resulting conflicts.
4. Immediately interrupt the user, regardless of his or her current activity, and ask what to do.
5. Defer the decision until the user is available to confer with.

This decision will depend on a number of features of the current situation, for example, how important the meeting is, how many conflicts it introduces into the schedule, what the user is currently doing, and so on.

Rather than try to develop handcrafted rules for determining which autonomy decision to make, we are attempting to use reinforcement learning (RL). Our approach is similar to the one taken in (Rudary, Singh, and Pollack 2004), in which we used RL to compute reminder policies, which map from features of the environment to decisions about whether and when to issue reminders. A key difference is that the reward signal for the autonomy case is less clear. We are currently investigating using a function of the user's subsequent actions (e.g., undoing a scheduling decision) and the quality of the resulting schedule. We are also planning to integrate the traditional unsupervised learning of RL with some advisability techniques, so that the user can focus the policy search by providing advice, e.g., "I wished you'd have asked me about that meeting before scheduling it." We are using direct policy search, because it defines a compact policy space that is generally interpretable by humans, and we are exploring a number of different policy gradient approaches from the RL literature.

Conclusions and Future Work

The concept of a persistent useful life motivates the design of PTIME. It has an extended notion of collaboration, between agents and more specifically with the individual user. The collaborative scheduling process is separated from the constraint reasoning algorithms to enable interaction with the user and other PTIME agents. This interaction forms the framework for learning and adjustable autonomy. The time management processes are represented as context-sensitive, hierarchical procedures. These provide hooks into the user's decision process at multiple levels of abstraction. These hooks can be used to passively learn the user's preferences or to facilitate the specification of advice from the user. The resulting agent will let the user retain control of decisions when necessary and relinquish control to the assistant at other times. Meanwhile, the agent will be sensitive to the user's wishes and preferences.

Future studies of the interactions between agents will be conducted in collaboration with USC-ISI, building on work by (Tambe and Zhang 2000). Meanwhile, the interaction between the agent and the human user will be studied in collaboration with the University of Michigan, based on previous work by (Pollack et al. 2003).

Acknowledgments. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DARPA or the Department of Interior-National Business Center (DOI-NBC).

References

- Berry, P.M., Gervasio, M., Uribe, T., Myers, K., and Nitz, K. (2004). A personalized calendar assistant, *In proceedings of the AAAI Spring Symposium Series*, Stanford University.
- Babaian, T., Grosz, B. and Shieber, S.M. (2002). A writer's collaborative aid. *In proceedings of the Intelligent User Interfaces Conference*, San Francisco, CA. January 13-16. ACM Press, pp. 7-14
- Bistarelli, S., Montanari, U., and Rossi. F. (2001). Solving and learning soft temporal constraints: Experimental scenario and examples, *In proceedings of the CP'01 Workshop on Modelling and Solving Problems with Soft Constraints*.
- Cheadle, A.M., Harvey, W., Sadler, A.J., Schimpf, J., Shen, K. and Wallace, M.G. (2003). *ECLiPSe: An Introduction*, Technical Report IC-Parc-03-1, IC--Parc, Imperial College London.
- Dechter, R., Meiri, I., and Pearl. J. (1991). Temporal constraint networks. *Artificial Intelligence*, 49(1–3):61–95.
- Franzin, M. S., Freuder, E. C., Rossi, F., and Wallace, R. (2002). Multi-agent meeting scheduling with preferences: Efficiency, privacy loss and solution quality. *In proceedings of the AAAI 2002 Workshop on Preference in AI and CP*.
- Grosz, B. and Kraus, S. (1999). The evolution of SharedPlans. In *Foundations and Theories of Rational Agencies*, A. Rao and M. Wooldridge, eds. pp. 227-262.
- Morley, D. (2004). *Introduction to SPARK*. Technical Report, Artificial Intelligence Center, SRI International, Menlo Park, CA.
- Myers, K. L. and Morley, D. N. (2003). Policy-based Agent Directability. In *Agent Autonomy*, Kluwer Academic Publishers.
- Payne, T. R., Singh, R., and Sycara, K. (2002). Rcal: A case study on semantic web agents, *In proceedings of the First International Conference on Autonomous Agents and Multi-agent Systems*.
- Peintner B. and Pollack, M.E. (2004). Low-cost addition of preferences to DTPs and TCSPs, In *AAAI-2004*, pages 723-728.
- Pollack, M.E., Brown, L., Colbry, D., McCarthy, C.E., Orosz, C., Peintner, B., Ramakrishnan, S., and Tsamardinos, I. (2003). Autominder: An intelligent cognitive orthotic system for people with memory impairment, *Robotics and Autonomous Systems*, 44:273-282, 2003.
- Pragnesh J.M., Shen W., Tambe M., and Yokoo, M. (2003). An asynchronous complete method for distributed constraint optimization, *In proceedings of the Autonomous Agents and Multi-Agent Systems, (AAMAS)*.
- Pragnesh, J.M., and Veloso, M. (2004). Multiagent Meeting Scheduling with Rescheduling, *In proceedings of the Fifth workshop on Distributed Constraint Reasoning (DCR 2004)*.
- M. Rudary, M., Singh, S., and Pollack, M.E. (2003). Adaptive Cognitive Orthotics: Combining Reinforcement Learning and Constraint-Based Temporal Reasoning, *In proceedings of the 21st International Conference on Machine Learning*, July.
- Stergiou, K., and Koubarakis, M. (1998). Backtracking algorithms for disjunctions of temporal constraints, *In proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, p.248-253, Madison, Wisconsin, United States.
- Tambe, M., and Zhang, W. (2000). Towards flexible teamwork in persistent teams:extended report, *Journal of Autonomous Agents and Multi-agent Systems*, Vol. 3 159-183.
- Tsamardinos, I., and Pollack, M.E. (2003). "Efficient Solution Techniques for Disjunctive Temporal Reasoning Problems," *Artificial Intelligence*, 151(1-2):43-90.