Mixed-Initiative Issues for a Personalized Time Management Assistant

Pauline M. Berry, Melinda T. Gervasio, Tomás E. Uribe, Neil Yorke-Smith

Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025
{berry,gervasio,uribe,nysmith}@ai.sri.com

Abstract

This paper explores the mixed-initiative issues arising in the Personalized Time Manager (PTIME) system. PTIME is a persistent assistant that builds on our previous work on a personalized calendar agent (PCalM) (Berry et al. 2004). In order to persist and be useful, an intelligent agent that includes collaborative human/agent decision processes must learn and adapt to the user's changing needs. PTIME is intended to support a richer dialogue between the user and the system, which should be useful to both. If the system can reliably lean the user's preferences and practices, trust between user and assistant will be established, decreasing the system's reliance on mundane user interaction over time. The enabling technologies include soft constraint satisfaction, multicriteria optimization, a rich process framework, learning, and advice.

Introduction

The human time management problem is intensely personal. Many people—especially busy workers—are reluctant to relinquish control over the management of their own time. Moreover, people have different preferences and practices regarding how they schedule their time, how they negotiate appointments with others, and how much information they are willing to share when doing so. They also have different needs and priorities regarding the reminders they should receive.

We are developing the Personalized Time Manager (PTIME) assistant, with the goal of managing an individual's temporal commitments in a consistent, integrated framework over an extended period of time, while recognizing the differences between individuals and adapting to these differences. The interaction between the human user and the system is central to this goal. To maximize the continued usefulness of this interaction, both the user and the system should benefit from it. The scheduling solutions found by the system should be informative and proactive, and the dialogue should improve the quality of future interactions.

The PTIME project is part of a larger, ambitious automated assistant called CALO. CALO is a cognitive assistant that supports its human user in a variety of ways. For example, project and task management, information collection, organization and presentation and meeting understanding. However, the focus of CALO is its ability to learn and persist. Our hypothesis is that for mixed-initiative systems to succeed in the long term, the dialogue between human and system must evolve over time. To achieve this, we are designing PTIME so that

1. PTIME will unobtrusively learn user preferences, using a combination of passive learning, active learning, and advice-taking;
2. As a result, the user will become more confident of PTIME's ability over time, 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.

Background

Tools and standards for representing, displaying, and sharing schedule information have become common. A generally adopted standard for calendar representation is iCalendar (RFC2447).

There are also many calendar tools to organize, display, and track commitments. However, most people still spend a considerable amount of time managing the constant changes and adjustments that must be made to their schedules. Desktop tools have dramatically improved the administration of our calendars, but their scheduling capabilities are limited. Automated meeting scheduling assistants have shown promise, but their use tends to be fleeting, since they do not evolve over time. People also use a variety of other tools, such as *to-do* lists, to keep track of workload and deadlines not supported in the typical calendar tools.

The emphasis in the research community has been on automated meeting scheduling: finding feasible time slots for meetings given a set of requirements on participants, times and locations. Work in this area can be generally divided into *Open* and *Closed* scheduling systems (Ephrati et al. 1994). In *Open* systems, individuals are autonomous, and responsible for creating and maintaining their own calendar and meeting schedules, perhaps selfishly. They can operate in an unbounded environment without constant obligation to one organization. In a *Closed* system, the meeting mechanisms are imposed on each individual, and a

consistent and complete global calendar is maintained. Closed systems are more common because preference measures can be normalized across users, participant availability is known at all times, and the problem can be formulated as constraint optimization. Not all closed systems are centralized, and there is interesting work in distributed solutions to the closed scheduling problem (Ephrati et. al.1994, Sandip and Durfee 1998).

Closed systems are rarely adopted because the users seldom live in a truly closed environment, and need to retain more personal control of their calendars. Open scheduling systems pose additional challenges, such as privacy: an individual may not wish to share all, part, or any of his schedule, or may choose not to participate in a meeting, but not divulge this information.

CALO exists in an open, unbounded environment where issues of privacy, authority, cross-organizational scheduling, and availability of participants abound. PTIME is similar in approach to RCAL (Payne et. al. 2002) but extends the notion of collaboration with the user. The scheduling task is viewed as a shared goal of the user and the agent. 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. PTIME considers finding the best solution as a dialogue between user and agent, and treats the underlying scheduling problem as a soft Constraint Satisfaction Problem (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.

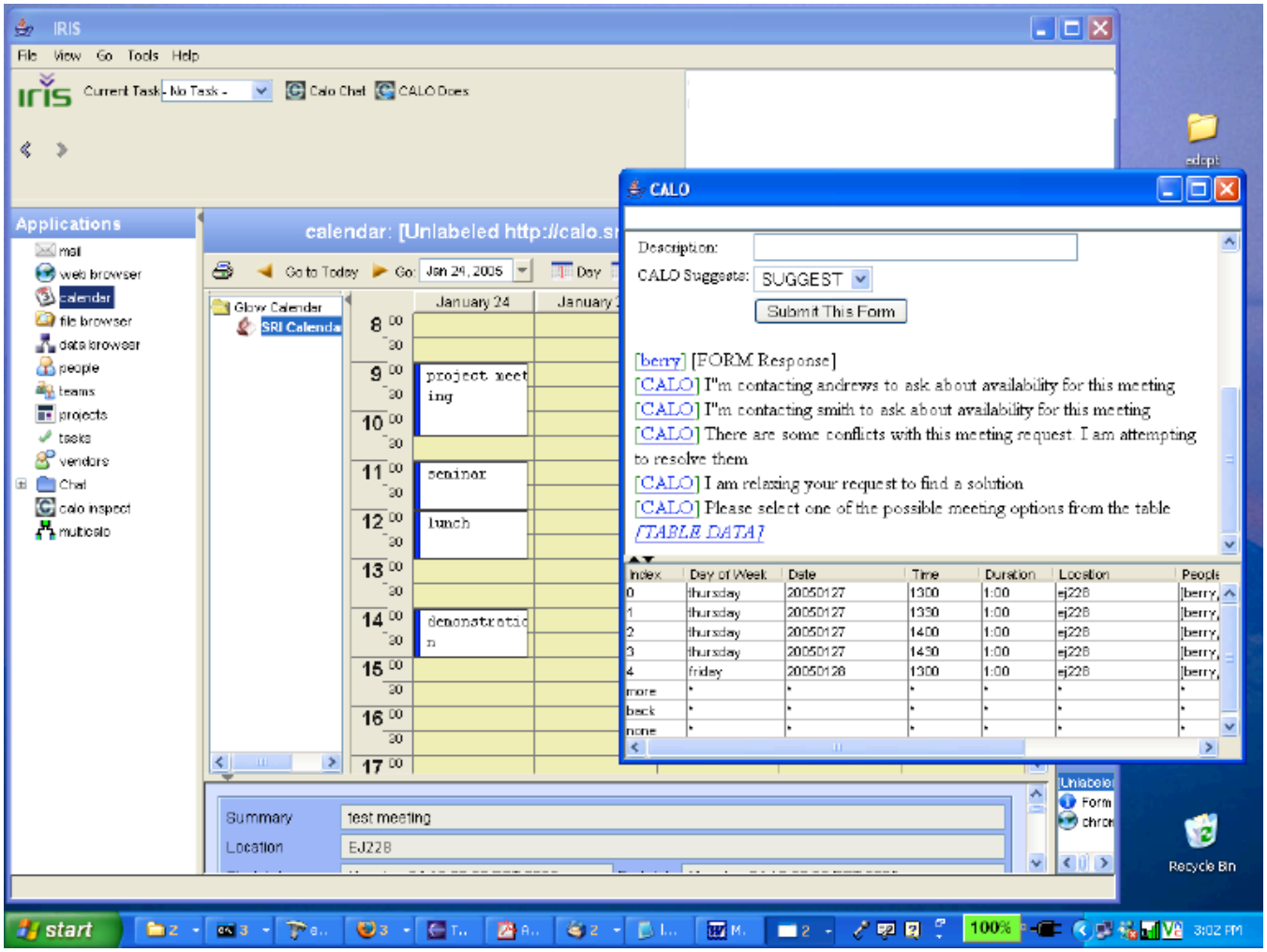

Figure 1: A screenshot from PTIME.

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-Control),** 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. Currently, we have developed PLIANT to learn temporal scheduling preference, e.g. *time of day, day or week, fragmentation of schedule*.
- **Advisability (PTIME-Control),** 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-Engine**), which permits reasoning within a unified plan representation. The representation used by PTIME unifies temporal and non-temporal constraints, soft and hard constraints, and preferences. The constraint reasoner (PTIME-Engine) considers workload issues and task deadlines when scheduling typical calendar events, such as meetings. The PTIME-Engine 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, Tsamardinos and Pollack 2003), to address complex constraint space and preference handling, and to enable partial constraint satisfaction. The PTIME-Engine can also explore alternative conflict resolution options via relaxation, negotiation, and explanation techniques, (Junker 2004).
- **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 learning algorithms to create reminders that are context-sensitive and personalized.
- **Adjustable Autonomy (PTIME-Control)**, 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.

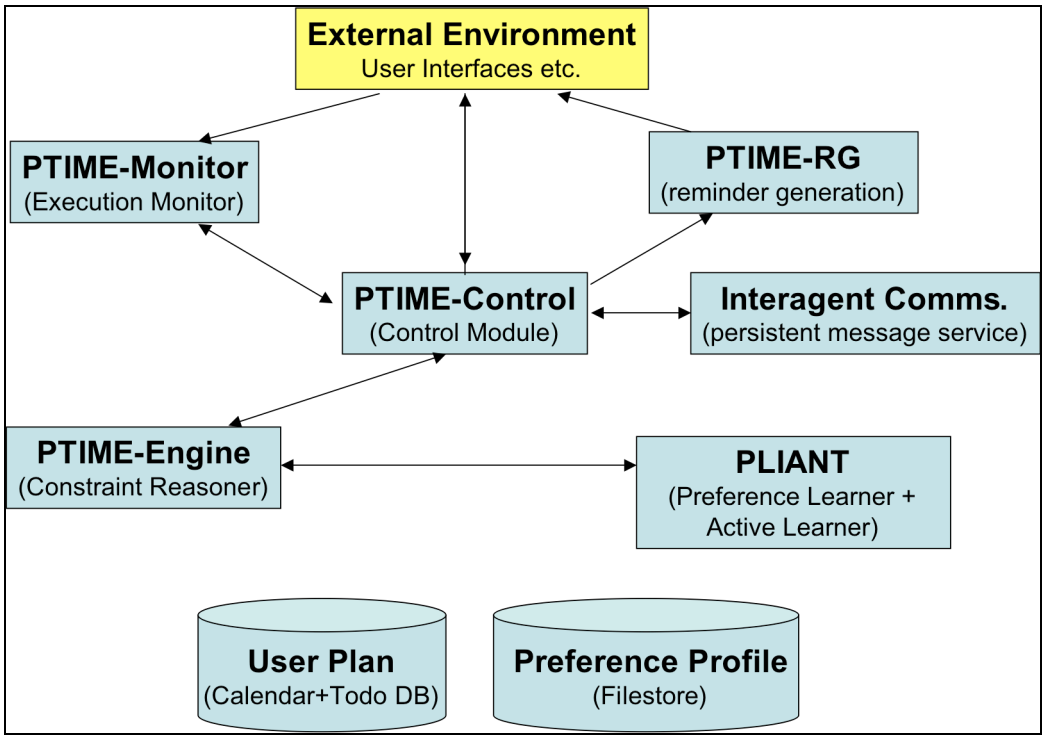

Figure 2: PTIME functional architecture

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:

(1) Scheduling: 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).

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

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 learned passively 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 schedule features (e.g., day of week, start time, fragmentation) (Gervasio et al. 2005). The features were selected to capture the temporal characteristics of a scheduling decision. 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 the problem of 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 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.

Mixed-Initiative 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. This section describes our research into hybrid constraint satisfaction, partial constraint satisfaction with preferences, negotiation and advice-taking. The result is a framework for negotiation between agents and with the user. We will also describe our ongoing work to learn preferences and accept advice from the user.

Using Preferences in Scheduling

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.

Critical to the mixed-initiative goals of PTIME is the ability to use the learned knowledge of user preferences within the underlying constraint satisfaction problem. (Berry et al. 2005) describes our approach to constraint satisfaction for PTIME, which involves a combination of disjunctive and finite-domain constraint solvers with preferences. Since it is relevant to this discussion, we now briefly discuss the representation of schedule preferences and relaxations within soft CSPs.

User preferences are mapped into the shape and height of specific preference functions for each of the relevant soft 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).

For example, suppose a meeting with Bob must occur before a seminar. If $S_M(S_S)$ and $E_M(E_S)$ are the start and end of the meeting (resp. seminar), the constraint is $c: E_M - S_S \leq 0$. Figure 3 shows the shape of the preference functions on c from left to right, if the constraint is hard, a little relaxable, and very relaxable.

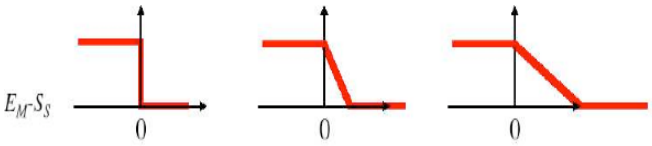

Figure 3: Example Preference Functions

To solve soft CSPs that include such preference functions, we combine existing solvers for the temporal and non-temporal constraint subproblems in a hybrid formulation. Constraint solving in PTIME is implemented in ECLiPSe (Cheadle et al. 2003), and is also compatible with SICStus Prolog; such Constraint Logic Programming systems are well-suited to hybrid solving. We are exploring search techniques that can produce not only a single optimal solution, but also a good set of qualitatively different solutions to present to the user. A good set of solutions has three characteristics: to include the most desirable solution, to give the user qualitatively different choices and to promote future learning.

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 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 of the processes within the context of the dialogue.

Figure 4 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 5 illustrates an example process in SPARK-L. Each decision point offers the choice to automate the decision, ask the user for advice or decision, postpone the decision, or take another action. For example, when the goal is:

[do: (select_solution $resultset $result)],

a set of different actions might be intended, including asking the user to select an option or automatically selecting the highest valued one. 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.

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 4. 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_conflct $constraints $result)]
           [do: (confirm_meeting $result $attributes)]]
  }
```

Figure 5. Example SPARK-L process

Advice

The PTIME-Controller can take user advice and conform to organizational policies. Advice is defined as an enforceable, well-specified constraint on the performance or application of an action in a given situation. In general advice can be considered to be a type of policy, often

personalized. (Sloman 1994) defines two types of policy: authorization and obligation. For our advisable system, we extend this categorization to include preference:

1. *Authorization* defines the actions that the agent is either permitted or forbidden to perform on a target.
2. *Obligation* defines the actions that an agent must perform on a set of targets when an event occurs. Obligation actions are always triggered by events, since the agent must know when to perform the specified actions.
3. *Preference* defines a ranking in the order or selection of an action under certain conditions.

Advice can both apply to the application of strategies, the conditions under which a strategy is applicable, or the instantation of a variable. Advice may be conflicting, can be long-lived, and their relevance may decay over time. Advice can be used to influence the selection of procedures and strategies for problem solving and also to influence adjustable autonomy. The management of advice is an active research focus for the CALO project. The application of advice is central to both PTIME for influencing preference and for controlling adjustable autonomy strategies.

Summary

The concept of a persistent useful interaction motivates the mixed-initiative design of PTIME. It has an extended notion of collaboration with the user, which forms the framework for learning and adjustable autonomy. The time management process is represented using context-sensitive, hierarchical procedures, which provide hooks, via the structured decision points, 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.

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 authors 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.

Berry, P.M. Gervasio, M., Uribe, T., and Yorke-Smith, N. (2005). *Multi-Criteria Constraint Solving and Relaxation for Personalized Systems*, Technical Report, SRI International.

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.

Ephrati, E., Zlotkin, G., and Rosenschein, J.S. (1994). A non manipulable meeting scheduling system, *In proceedings of the Thirteenth International Distributed Artificial Intelligence Workshop*, Seattle.

Gervasio, M.T., Moffitt, M.D., Pollack, M.E., Taylor, J. and Uribe, T.E. (2005). *In proceedings of the International Conference in Intelligent User Interfaces (IUI),* San Diego.

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.

Junker, E. (2004). QuickXplain: Preferred Explanations and Relaxations for Over-Constrained Problems. *In proceedings of AAAI-04*.

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 proceedings of AAAI-04*, 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.

Sandip, S., and Durfee, E.H. (1998). A formal study of distributed meeting scheduling. *Group Decision and Negotiation*, vol. 7, pp. 265-298.

Sloman, M. (1994). Policy driven management for distributed systems. *Plenum Press Journal of Network and Systems Management,* vol.2, no. 4, pp. 333-360.

Stergiou, K., and Koubarakis, M. (1998). Backtracking algorithms for disjunctions of temporal constraints, *In proceedings of AAAI/ IAAI-98*, p.248-253, Madison.

Tsamardinos, I., and Pollack, M.E. (2003). Efficient Solution Techniques for Disjunctive Temporal Reasoning Problems, *Artificial Intelligence*, 151(1-2):43-90.