To appear in the IEEE Intl. Symposium on Computational Intelligence in Robotics and Automation, 1999

Towards Focused Plan Monitoring: A Technique and an Application to Mobile Robots

Martha E. Pollack[*,†] and **Colleen McCarthy**[*]
Department of Computer Science[*] and Intelligent Systems Program[†]
University of Pittsburgh
Pittsburgh, PA 15260
{pollack,colleen@cs.pitt.edu}

Abstract

Until recently, techniques for AI plan generation relied on highly restrictive assumptions that were almost always violated in real-world environments; consequently, robot designers adopted reactive architectures and avoided AI planning techniques. Some recent research efforts have focused on obviating such assumptions by developing techniques that enable the generation and execution of plans in dynamic, uncertain environments. In this paper, we discuss one such technique, *rationale-based monitoring*, originally introduced by Veloso, Pollack, and Cox [9], and describe our use of it in a simple mobile robot environment. We review the original approach, describe how it can be adapted for a causal-link planner, and provide experimental results demonstrating that it can lead to improved plans without consuming excessive overhead. We also describe our use of rationale-based monitoring in a mobile robot office-assistant project currently in progress.

1 Introduction

Until recently, techniques for AI plan generation relied on highly restrictive assumptions that were almost always violated in real-world environments; consequently, robot designers adopted reactive architectures and avoided AI planning techniques [1]. Some recent research efforts have focused on obviating such assumptions by developing techniques that enable the generation and execution of plans in dynamic, uncertain environments. In this paper, we discuss one such technique, *rationale-based monitoring*(RBM), originally introduced by Veloso, Pollack, and Cox [9] (hereafter VPC), and describe our use of it in a simple mobile robot environment.

RBM is intended to help plan-generation systems perform effectively in dynamic environments. As is well-known in the robotics community, changes in such environments may occur during plan generation and execution. The motivation behind RBM is to identify those features of the environment that are most likely to impact the plan being formed or executed. The RBM approach takes seriously the view that planning is a decision-making process. Thus, in the RBM framework the rationale behind each plan decision is recorded so that attention can be focused on those conditions that are most likely to change a planning decision.

The authors of the original paper on RBM described three processing stages: (1) generating monitors that represent environmental features that are potentially relevant to the plan; (2) deliberating, whenever a monitor fires, about whether to respond to that monitor; and (3) transforming the plan as warranted. They also described a preliminary implementation of their approach in the Prodigy planning system. In this paper we adapt RBM for a causal-link planner, provide experimental results to demonstrate its feasibility, and discuss its use in a mobile-robot office-assistant project currently in progress.

2 RBM for Robots

Mobile robots inevitably need to track changes in their goals and their environments; rationale-based monitoring provides a potential way of focusing their monitoring, assuming the robot agent is plan-based. Consider, for example, a mobile robot of the familiar "office gopher" type, whose tasks range from delivering coffee, mail, and faxes to planning entire meetings and making arrangements for meals and accommodations. Such a robot would require the ability to generate complex plans beyond the reactive level, and to adjust those plans as its environment changes.

In general, fully interleaved planning and execution would be desirable. However, existing techniques for full interleaving do not address the problem of automatically generating focused monitor sets. [1] RBM, as so far developed, facilitates monitoring *during* plan generation, but not (yet) during plan execution. VPC note that many planning problems take place over a period of hours to days, including, for example, planning a vacation or a large-scale military operation. During extended planning, new information that affects plan decisions may arrive and should be attended to.

[1] Most reactive planning systems [5; 4; 3] rely on hand-encoding of monitoring actions.

In the mobile robot scenario, we view the robot's operation as being divided into sequential periods of time, where the optimal period for the time segments needs to be determined empirically in the target domain. (Indeed, the segments need not all have the same duration: at night, for example, they may be much longer than during the day.) While the robot is performing its activities during period *i*, it is also forming plans for future periods (*i+1, i+2, ...*). During period *i*, it can also accept requests for new tasks for future periods, and similarly, can receive information about actual and anticipated world changes that may impact the plans it is forming for future periods.[2] Thus the robot might initially form a plan for the next period to service orders to the third floor before the second. If it learns that elevator use will be restricted during the next period, it might change the order of the actions in its plans. If the restriction is then reversed, it might reconsider its initial plan. In the extreme case, attending to relevant changes may mean the difference between valid and invalid plans. If the robot learns that the the elevator will be completely out of service at the relevant time, then its initial plan may be useless unless modified in response to this new condition. Of course, changes such as these may also occur *during* plan execution, and we hope in future work to extend the ideas behind RBM to full interleaving of planning and execution. For now, however, it is useful to have a principled technique to enable plan generation to proceed in dynamic environments.

3 An Overview of RBM

As mentioned earlier, RBM is based on the idea of planning as decision making. In an RBM framework, the rationale behind each decision is recorded and used as a basis for establishing monitors. There are clearly connections between the RBM approach and techniques for execution monitoring that date all the way back to the development of triangle tables [2], originally used in the Shakey the robot project at SRI. RBM generalizes earlier forms of monitoring, not only by making monitoring relevant to plan *generation*, but also by considering a wider set of types of monitor and of planning responses.

3.1 State Dependent Planning Decisions

An RBM planner must keep track of the planning decisions that depend on the current world state, and thus are subject to change. All planners make certain types of decisions: they decide which actions to employ in a plan (action selection), which objects to apply those actions to (parameter binding), and which new subgoals are introduced by a particular action (subgoal introduction). VPC identified three types of monitors, representing world-state conditions that can influence planning decisions, and we have added a fourth (preference-condition monitors).

[2]Real-time sensing will also be required to make critical changes to plans underway for the current time period, but, at least for now, that would be handled by a different mechanism, perhaps involving hand-encoding of reactive control methods.

Subgoal monitors encode all the preconditions and bindings of operators that have been considered so far in the planning process. The truth value of these conditions can influence decisions about action selection, parameter binding, and subgoal introduction. For instance, a particular choice of action and parameter binding may be preferred because the action's preconditions are currently true under that binding.

Usability-condition monitors represent any usability constraints for operators considered so far in the planning process. In contrast to preconditions, usability conditions are intended to represent conditions that the agent either cannot or should not modify. The truth value of usability conditions can influence decisions about action selection and parameter binding: a particular action/parameter binding combination may be deemed impossible, if the action's usability conditions are currently false under that binding. UCPOP [8] and related causal-link systems have typically not supported usability conditions, but because they are a useful construct, we implemented them for our experiments and our mobile robot application.

Quantified-condition monitors are derived from the universally quantified preconditions in any operator considered during planning. These monitors track the extension of the universally quantified condition. For instance, if a plan includes a step of delivering a memo to all employees in a certain group, it is important to know if someone has been added to or deleted from the group. Although quantified-condition monitors are clearly important and interesting, VPC only briefly mention their implementation, and we have not yet implemented them.

Preference-condition monitors represent user-specified preference information, which may be independent of preconditions or usability conditions. Suppose the robot has an operator for supplying food from the cafeteria for a meeting. While both doughnuts and sandwiches are acceptable, the user might specify a preference for doughnuts in the morning and sandwiches in the afternoon. The truth of the condition "meeting-time $<$ noon" will then influence action selection, but it is neither a precondition nor a usability condition for supplying food. In the UCPOP system, such preference information can be directly implemented with user-specified search control rules.

3.2 Monitor Generation and Response

To include rationale-based monitoring in a causal-link planner, two steps must be taken. First, monitors must be generated. This occurs throughout the planning process when new planning nodes are generated. Whenever a partial plan is extended by the addition of a new step, a monitor is generated for every precondition and usability condition associated with that step. The current truth value of the monitored condition is stored, along with a pointer to the node(s) in the search space that rely on that condition. In addition, whenever a search control rule that relies on a domain condition is triggered, a monitor is established for that condition.

Second, when a condition being monitored changes, an

Loop until a solution is found
1. For each monitor that fired since the last iteration
2. Update the current state information.
3. For each affected node N
4. Make needed revisions to N.
5. Recompute the ranking function for N.
6. Select the next node for expansion, M.
7. Make M consistent with the initial state.
8. Expand M.

Figure 1: The Top-Level Algorithm

appropriate response must be taken. More specifically, when a monitor for some condition C fires, indicating that the truth value of C has changed, the affected nodes must be reassessed and revised as necessary.[3] The revisions to the plan depend upon the type of monitor and the polarity of change (whether a true condition became false or vice versa). For instance, consider the case in which a subgoal monitor indicates a condition C has become true. In the partial plans that include steps with precondition C, any steps that only support the establishment of C may be removed and replaced with a link from the dummy initial step.

Plan reassessment is straightforward in the causal-link framework—indeed, much simpler than in the Prodigy implementation of VPC. Recall that causal-link planners perform a best-first search. At each iteration, they employ a node-selection function, also called a ranking function, to determine the current best partial plan, which is then selected for subsequent expansion. Typically, node selection is a function of the number of steps already in the plan, the number of open (unestablished) preconditions, and, sometimes, the number of threats [6]. However, as mentioned above, user-defined search control rules that rely on domain conditions can also be employed. When a monitor fires during planning, the ranking function must be recomputed for the affected nodes. The node with highest recomputed rank can then be selected for subsequent expansion, as is standard in best-first search.

4 Implementation in UCPOP

We implemented RBM in the UCPOP causal-link planner [8], and used this in our mobile robot application. Figure 1 provides the top-level algorithm. As can be seen, generating and responding to monitors is tightly interwoven into the algorithm's search process.

Two key data structures are added to the basic UCPOP system. The first simply records information about the current state. Whenever a monitor fires this state information is updated. The second added data structure is the monitoring hash table, which records monitored conditions, their current truth values, and pointers to the nodes that are relevant for each monitored condition. For subgoal, usability-condition, and quantified-condition monitors, a node is relevant if it has the monitored condition as a link or flaw. All nodes are considered to be relevant to a preference-condition monitor.

During planning, the first step is to make any updates needed as a result of changes in the monitored conditions. For each monitor that has fired, indicating a change in condition C, the current state information must be updated (line 2), and all nodes relevant to C must be revised (lines 4-5). The revisions made depend on the type of monitor and the polarity of change, as summarized in Figure 2. Note that in all cases, the ranking function must be recomputed for the relevant nodes. Plan expansion can then occur: the next node is selected (line 6) and its successors generated (line 8). Prior to expansion, however, the selected node must be checked because its initial state (i.e., the effects of its dummy initial step) may need to be updated to be made consistent with the true current state. To see why this is required, suppose that at some point during planning, the monitor for some condition C fires, and assume that N is a node that is not relevant for C. Then N will not be immediately updated: updates in lines 4-5 will only apply to nodes that are relevant for C. Suppose that later N is selected for expansion. It is possible that one of its successor nodes will be relevant for C, for instance because it contains an operator with C as a precondition. To ensure that the successor nodes inherit correct information about the initial state, N must be made consistent with the true current state when it is selected for expansion.

For clarity, we have written the algorithm as if it terminates as soon as a solution is found. In practice, whenever a process instantiating the planning algorithm is invoked, it should remain alive until the beginning of the time segment that includes execution of the generated plans. That way, changes that occur after plan generation, but prior to execution, can be handled appropriately.

5 Feasibility Experiments

To explore the overhead of RBM, we conducted controlled experiments in which we had the planner generate plans both with and without RBM, varying the amount of change in the environment. The experiments were all conducted on a Sparc Ultra 1 workstation,using UCPOP's built-in data collection routines to gather timing results. We simulated changes to the environment by creating a file that is read on each planning cycle: each line represents changes that have occurred since the previous iteration. We performed initial experiments using various standard planning domains, including those in the UCPOP distribution suite and a set of vacation-planning problems that we encoded to investigate preference rules. In this paper, we focus for space reasons on results from the MCD-Grid-World taken from the UCPOP suite. Results in other domains were consistent with the results reported here.

[3] In this paper we assume that sensor values for the monitored conditions have already been converted to truth values.

Monitor Type	Polarity of Change	Revision
Subgoal	T → F	Update initial state; open the monitored condition; recompute h.
	F → T	Update initial state; remove steps and links that only establish the monitored condition; recompute h.
Usability	T → F	Update initial state; mark node as impossible.
	F → T	Update initial state; remove mark as impossible; recompute h.
Preference	T → F	Update initial state; recompute h.
	F → T	Update initial state; recompute h.

Figure 2: Revisions after Fired Monitors

In the main experiment, we ran the planner, with and without monitoring on six different problems from the MCD-Grid-World. The solutions to these problems range from two to ten steps long. With monitoring, we varied the frequency of (simulated) change in the world to model both rapidly changing and slowly changing environments: we considered cases in which a change occurs in 0%, .05%, .1%, 1%, 5%, and 50% of the planning iterations, with monitoring occurring at every cycle.[4] We generated between five and twenty different input files for each condition, depending on plan length and rate of change; we report the average times taken by the planner. To summarize, there are 42 experimental conditions: six problems, each run first without monitoring and then with monitoring for six different rates of change.

5.1 Case I: Identical Plans

Clearly, when the world never changes, the use of monitoring will not alter the plan that is generated. However, even when the world is changing, the changes that occur may not affect the validity or quality of the plan being formed. Then the planner will generate the same plan regardless of whether it uses monitoring.

Figure 3 plots the mean time taken by the planner without monitoring, and with monitoring for each rate of world change, for those cases in which the plans produced were identical in all circumstances. The x-axis shows the length of the original plan, i.e., the one generated without monitoring. The y-axis, which is logarithmic, plots mean time to find a solution. As can be seen, RBM as implemented in the UCPOP system generally incurs very little computational overhead. We note, however, that this result can be affected by oscillating conditions. In a few cases, we saw the monitoring system taking much longer than average. Analysis of these cases reveals that they included conditions whose truth values oscillated frequently.

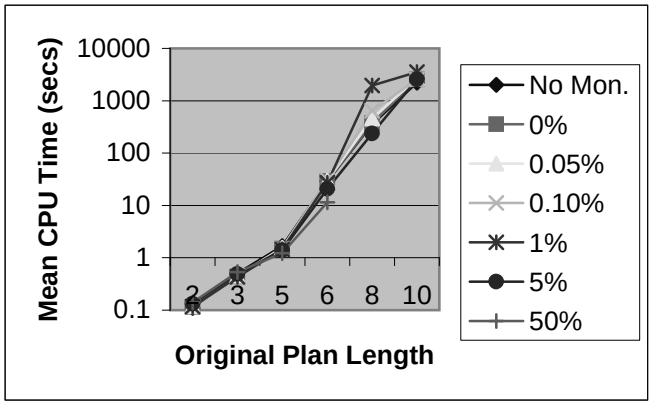

Figure 3: Identical Plans Produced

5.2 Better Plans

Sometimes the world changes in a way that allows the generation of an improved (shorter) plan than would otherwise be possible. In such cases, what overhead there is in monitoring may be outweighed by the time saved in generating a shorter plan. This is precisely what we saw in our experiments. Unfortunately, though, for the current domain, only a small number of the randomly generated cases fell into this category. We saw many more instances of monitoring leading to better plans in other domains, notably the TRAINS domains. In these other domains, monitoring again frequently took less time to find shorter plans than those found without monitoring.

5.3 Correct Plans

A more extreme case occurs when a change in the world "breaks" the plan being generated. In this situation, failure to monitor will lead to the generation of an incorrect plan. Monitoring prevents this problem, but it may require extra time to produce a plan. Figure 4 plots the average times to generate a plan with and without monitoring (averaging over all rates of change) . Sometimes, the additional time is required because the correct plan is longer than the incorrect one. For instance, the peak in Figure 4 for original plan length eight is due to the fact that in one problem the length of a correct plan requires 14 (and not eight) steps. Oscillating conditions again increased mean time for some of the problems in this category. Note that there is a data point missing for original plan length of three in the figure because none of the randomly generated input files led to this particular case.

[4] The input files representing the changes were constructed as follows: when simulating change p% of the time, for each line of the file we generate a random number n and if $n \geq p$, we randomly select a world-state condition C and include C and all its correlated conditions on that line. If $n < p$, we instead specify that there have been no changes during the previous iteration.

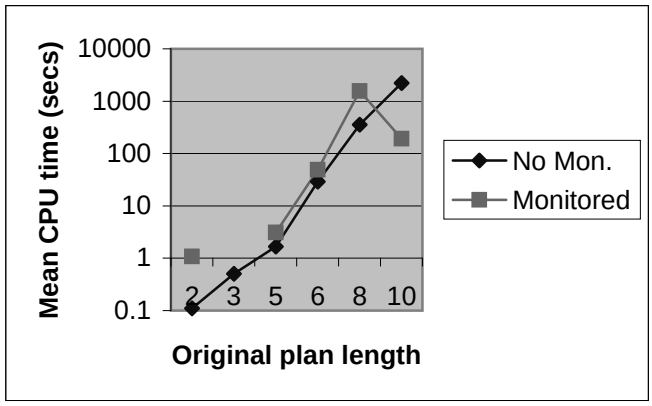

Figure 4: Correct Plans Produced with Monitoring

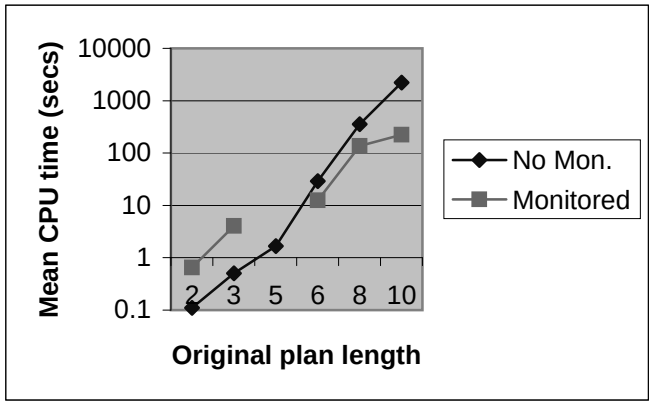

Figure 5: Failure Detected by Monitoring

5.4 Failure Detection

The last situation arises when a world-state change renders the original planning problem unsolvable. In such cases, monitoring will detect the problem, often fairly quickly. In contrast, planning without monitoring will continue to completion, generating a plan that is incorrect. Here monitoring is again required to produce a correct result; in addition, it usually achieves that result faster than planning without monitoring. Figure 5 shows that there is a break in the data because no cases of original length five led to failure under monitoring.

6 RBM in the Office-Gopher Domain

Having satisfied ourselves that the RBM approach appeared not to require excessive overhead, we incorporated it into a mobile robot office gopher. Office gophers inhabit dynamic environments, for which RBM is a potentially valuable technique: meeting times and locations are changed, elevators and computer systems have scheduled "down times", tasks are issued, retracted, reissued, and revised. Our robot, Rosie, is a Nomad Scout Beta 1.1 that resides in one of the computer science buildings at the University of Pittsburgh. She is currently run via a laptop running Windows NT, mounted on her top surface. Plan generation is performed on a

```
(:operator pickup
    :parameters ((object ?x) (location ?loc))
    :precondition (:and (:neq ?x robot) (at ?x ?loc)
        (at robot ?loc))
    :effect (grasping ?x))

(:operator drop
    :parameters ((object ?x))
    :precondition (:and (:neq ?x robot) (grasping ?x))
    :effect (:not (grasping ?x)))

(:operator move
    :parameters ((location ?from) (location ?to))
    :precondition (:and (:neq ?from ?to) (at robot ?from)
        (connected ?from ?to))
    :effect
    (:and (at robot ?to)
        (:not (at robot ?from))
        (:forall (?x)
            (:when (:and (grasping ?x) (object ?x))
            (:and (at ?x ?to) (:not (at ?x ?from)))))))
```

Figure 6: Example Operators

Sparc 1 and the resulting plan is interpreted and transmitted to the laptop, which then issues low-level sonar and motion commands to Rosie via a serial port.

Rosie operated in several different environments that we designed. A typical environment had seven rooms connected by single hallway and various items (coffee, copies, mail) positioned in different rooms. Figure 6 shows our encoding of the standard robot operators: **Move**, **Drop**, and **Pick-Up**(Figure 6). **Move** is translated into low-level robot code during planning. **Pick-Up** and and **Drop** are simulated in our current experiments, since Rosie is not equipped with manipulators. Additional operators include **Prepare Meeting** and **Supply-Food**. A example of a preference rule is one that expresses the preference for serving different types of food at different times of the day.

For our initial experiments, we divided Rosie's "day" into only two segments: night (6pm to 8am) and day (8am to 6pm). We assume that most tasks will be performed during the day. Because there will presumably be few tasks carried out at night, Rosie can also use that period to perform active sensing of its environment. This design is reminiscent of that used by Greenwald and Dean for planning gate usage at airports [7].

We imagine that users (office workers) will typically submit their their task requests for the next day before leaving for the night, and planning begins. Subsequently, the users can submit changes to those tasks, as well as to request new changes. For instance, the user may decide to change the start time for a meeting, thus requiring the plan to be revised: not only will the time change, but it may need to be held in a different room, and the food to be served may change. Information can also be input to the system about general world changes, such

as a scheduled elevator outage.

UCPOP supplemented with an RBM mechanism, as described in Section 4, was able to generate correct plans in times ranging from 2.3 seconds to over 5 hours. All plans generated were actually run on Rosie to verify correctness. In a typical situation, Rosie is given the initial state of rooms (free or busy), the presence or lack of elevator availability, and the locations of food, and is then asked to prepare a morning meeting. During planning time, the initially unavailable elevator was made available, and the time for the meeting was changed from the morning to the afternoon. The next morning, Rosie executed a plan that incorporated the elevator and served doughnuts.[5]

This domain—and our robot—are clearly too simple along a number of dimensions. Nonetheless, we believe that these experiments provide an indication of the viability of RBM for real-world robotic domains. In the office gopher domain, events change enough to merit monitoring, yet it is not so volatile that oscillating conditions present a problem. Resource-bounded monitoring led to better plans when opportunities arose and the identification of unattainable goals when failures were imminent, saving both robot and human resources.

7 Conclusions

In the current paper, we have shown how rationale-based monitoring can be adapted to causal-link planning and applied in a mobile robot application. Using RBM, a planning system can often generate plans that are more efficient than those that would be produced without monitoring; in fact, sometimes the failure to monitor may produce incorrect plans that the robot would fruitlessly attempt to execute. We presented experimental results demonstrating the use of RBM does not necessarily lead to excessive overhead.

There are several important directions for future work. We observed that a key factor influencing the cost of monitoring is oscillating world-state conditions. It would thus be useful to refine the monitoring approach to recognize those conditions that are prone to oscillation. Another issue involves the frequency of monitoring. In the experiments reported here, monitored conditions were checked at every planning cycle. Experiments we are now conducting suggest that less frequent sensing may further increase efficiency. Perhaps the most important remaining question concerns generalizing the RBM framework to fully interleaved planning and execution, sometimes called "continuous planning". If interleaved, modifying a plan can have a substantial execution cost not reflected in the ranking function of the best-first search. We hinted at this above, with our example of changing the time of a meeting: if you have already ordered morning food, then the caterer may charge you a penalty for changing your order. A key research question then, is how to take account of the penalties for modifying a partially executed plan when deciding whether to respond to a fired monitor. If full interleaving is possible, then an additional topic, specific to robotic applications, concerns the the potential for RBM to alleviate the problem of localization. Plan generation might begin before an exact initial position is known, with the plan modified as sensing narrows in on more precise coordinates.

[5] Actually, Rosie executed a plan that involved moving to a spot with a "simulated elevator", which was actually the stairs (the CS building does not have an elevator). Similarly, because Rosie does not have a gripping device, it moved to the spot where the doughnuts are located and assumed a human assistant would put them on board.

Acknowledgments

This research has been supported by the National Science Foundation, grant IRI-9619579, by the Air Force Office of Scientific Research, Contract F49620-98-1-0436, and by fellowship stipend support from the National Physical Sciences Consortium and the National Security Agency.

References

[1] Ronald C. Arkin. *Behavior-Based Robotics* MIT Press, 1998.

[2] R. E. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3(4):251–288, 1972.

[3] R. James Firby. Task networks for controlling continuous processes: Issues in reactive planning. In *Proceedings of the Second International Conference on AI Planning Systems*, pages 49–54, 1994.

[4] Erann Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 802–815, 1992.

[5] Michael P. Georgeff and Felix F. Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 972–978, Detroit, MI, 1989.

[6] Alfonso Gerevini and Lenhart Schubert. Accelerating partial-order planners: Some techniques for effective search control and pruning. *Journal of Artificial Intelligence Research*, 5:95–137, 1996.

[7] Lloyd Greenwald and Thomas Dean. Solving time-critical decision-making problems with predictable computational demands. In *Proceedings of the Second International Conference on AI Planning Systems (AIPS)*, pages 25–30, Chicago, IL, 1994.

[8] J. Scott Penberthy and Daniel Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Proceedings of the Third International Conference on Knowledge Representation and Reasoning*, pages 103–114, Cambridge, MA, 1992.

[9] Manuela M. Veloso, Martha E. Pollack, and Michael T. Cox. Rationale-based monitoring for planning in dynamic environments. In *Proceedings of the Fourth International Conference on AI Planning Systems (AIPS-98)*, 1998.