Rationale-Based Monitoring for Planning in Dynamic Environments

Manuela M. Veloso, Martha E. Pollack,* and Michael T. Cox
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We describe a framework for planning in dynamic environments. A central question is how to focus the sensing performed by such a system, so that it responds appropriately to relevant changes, but does not attempt to monitor all the changes that could possibly occur in the world. To achieve the required balance, we introduce ***rationale-based monitors***, which represent the features of the world state that are included in the plan rationale, i.e., the reasons for the planning decisions so far made. Rationale-based monitors capture information both about the plan currently under development and the alternative choices that were found but not pursued. We discuss the plan transformations that may result from the firing of a rationale-based monitor, for example when an alternative choice is detected. We have implemented the generation of and response to rationale-based monitoring within the Prodigy planner, and we describe experiments that show the feasibility of our approach.

Introduction

Until recently, much of the work in AI planning has been governed by a number of simplifying assumptions, notably, that the target environment is *static* and *determinate*. By static, we mean that the environment does not change while a plan is being formed, or between the time the plan is formed and the time at which it is executed; moreover, during execution, the only changes are those that are due to the actions specified in the plan. By determinate, we mean that the planning agent knows all the relevant facts about its environment, and that the actions available to it all have definite outcomes. A third important assumption has been that the goals being planned for are *categorical*, i.e., they are either satisfied or not, but there is no notion of partial satisfaction.

Of course, in most real-world environments, these assumptions are invalid, and techniques have consequently been developed to relax some of them. Conditional planners (Peot & Smith 1992; Collins & Pryor 1995) remove the need to assume that the planning agent is omniscient, while probabilistic planners (Goldman & Boddy 1994; Kushmerick, Hanks, & Weld 1995) obviate the assumption that actions have deterministic outcomes. These techniques have been combined in conditional, probabilistic systems such as C-Buridan (Draper, Hanks, & Weld 1994) and Weaver (Blythe 1998). Utility-based and decision-theoretic planning systems, (e.g., (Drummond & Bresina 1990; Haddawy & Suwandi 1994; Williamson & Hanks 1996; Onder & Pollack 1997) move away from the assumption of categorical goals. Finally, the work on Markov decision processes, e.g. (Boutilier, Dean, & Hanks 1995), can be seen as aimed at addressing both the assumptions of determinacy and categoricity of goals.

Less work has been done on removing the assumption of a static environment. One approach has been to rely on techniques for planning under uncertainty, treating exogenous changes that can be partly anticipated in a manner analogous to the way in which uncertain actions are treated; this approach has been taken both within a traditional planning framework (Blythe 1998), and is also inherent in the MDP literature. Another body of work, sometimes called "reactive planning" (Firby 1994; Gat 1992; Georgeff & Ingrand 1989), addresses run-time system behavior, and deals with the problem of dynamic environments by supplementing high-level plans, such as those produced by classical planners, with mechanisms for translating those plans into low-level behaviors that are responsive to changes in the world.

In this paper, we describe work aimed at addressing planning in dynamic environments. We are concerned with changes in the world that occur *during planning* itself. Indeed, once the assumptions of classical planning are abandoned, it is often important to interleave planning and execution, and the separation between plan time and execution time disappears. But even when planning takes place distinct from and prior to execution, it may still be necessary to modify partial

*Visiting Professor from the Department of Computer Science, University of Pittsburgh.

plans in response to changes in the environment that are observed during planning. An example is planning a vacation or a large-scale military operation. Even though the planning may take place largely or even wholly in advance of the execution, the planning process itself may take anywhere from hours to days. During that time, the planners may become aware of changes in the world that affect the plans they are forming.

The challenge in designing planners for dynamic environments is to achieve the proper measure of sensitivity to changes in the environment. In general, it is too costly to be responsive to every environmental feature that the planning system knows about. The need to balance sensitivity to environmental change against appropriate stability of the plans being formed is strongly reminiscent of the ideas that led to the design of the IRMA architecture and filtering strategy (Bratman, Israel, & Pollack 1988).

To achieve the needed balance, we introduce a mechanism called *rationale-based monitoring.* Planning is strongly identified as a decision making process and the planning system itself records the rationale for the choices it makes. The planner then monitors only those environmental changes that would affect the truth-value of the planning rationale. We have implemented a version of rationale-based monitoring within the Prodigy system (Veloso *et al.* 1995), but the rationale-based monitoring technique could readily be applied to other planning systems as well. We describe controlled experiments that demonstrate the feasibility of rationale-based monitoring in dynamic environments.

Rationale-Based Monitoring

As just noted, the key idea underlying our work is to perform sensing during planning, so that changes in the world can influence the planning process. Sensing all the features of the world is impossible. Ideally, one would want to sense only the *relevant* or *potentially relevant* features. We introduce the idea of *rationale-based monitors*, which provide a means of focusing attention on features of the world likely to affect the plan. When a feature being monitored changes, we say that the monitor *fires*. Deliberation can then be performed to decide whether the plan under construction should be changed, and if so, in what way. There are thus three major steps to be performed in rationale-based monitoring:

- Monitor generation: Particular features of the world must be identified as potentially relevant to the planning process. In our approach, monitors are generated dynamically during plan generation, and follow directly from the plan rationale.
- Deliberation: When a monitor identifies a potentially relevant change in the world state, the planner needs to deliberate about whether the change warrants an alteration of the plan being constructed.
- Plan transformations: If the planner decides to attend to the detected changes of the world state, there are several different ways in which the plan may be transformed. In particular, parts of the plan may be deleted because they have become unnecessary; new goals may need to be added and current ones refined; and prior decisions about how to achieve particular goals may be changed.

In this paper, we focus on the first and third steps. We are currently developing methods for deliberating about whether to attend to a monitor that has fired.

Monitor Generation

The first stage of a rationale-based monitoring approach involves deciding which features of the world state to monitor. Planning in its essence is a decision making process, and we view the *plan rationale* as the *reasons* that support the planner's decisions. The exact decisions that a planner faces vary somewhat in form but little in content among different classical planning algorithms. Every planning algorithm must decide what actions to use to achieve its goals, what objects to apply each action to, and what new subgoals are introduced with a new action.

Influence of the World State in Planning Decisions

Each of the planning decisions may be influenced by the planner's current beliefs about the state of the world, as now described.

Action selection: Given a goal G, a planner needs to decide what operator to use to achieve G. In general, there may be several candidate operators, and a planner will create a different alternative plan for each. An alternative is selected for expansion according to the particular evaluation technique used.

Often, the decision to pursue one alternative over another will be highly dependent on the current world state. For example, assume that there are two operators O_1 and O_2 that can achieve some goal g. If *all* of the preconditions of O_1, and *none* of the preconditions of O_2 are satisfied in the world state, then a reasonable search-control evaluation would prefer the plan that uses O_1.[1] Different existing planning systems will encode this type of preference in different ways. For instance, in Prodigy it is embodied in the use of conspiracy numbers (Blythe & Veloso

[1]In this paper, we follow the practice of many planning systems and ignore the differential costs of different actions; in subsequent work we will remove this assumption. Note also that while the current example is extreme in assuming that *all* and *none* of the preconditions of the competing assumptions are satisfied, the basic principle holds in more general situations.

1992); in partial-order causal link (POCL) systems such as UCPOP (Penberthy & Weld 1992), it is a component of most node-selection strategies.

A second influence of the world state arises in those planning systems that distinguish between *preconditions*, which may be adopted as subgoals, and *usability conditions* (also known as "static preconditions"), whose achievability is outside the control of the planner. For instance, a planner for a transportation domain might have a usability condition for the operator "land helicopter in city C," specifying that the weather in C be clear. If a usability condition for an operator O is not true in the current state, then a plan including O will not be further expanded by the planner.

Step instantiation: A planner may need to decide what resources will be used in performing an action. This is typically implemented by means of parameter-binding, though in slightly different ways for different planners. In Prodigy, variable binding is a distinct decision that follows an action selection decision, to produce fully-instantiated preconditions. In POCL planners such as UCPOP, some binding decisions are made during operator selection (by means of unification with the goal and/or as a side-effect of step re-use), while other binding decisions occur when threats are resolved by separation. In either case, as with action selection, different binding decisions will lead to different partial plans, and the planning system must choose amongst these.

Again, this choice may be influenced by the planner's beliefs about the current world state. This can occur, for example, if the domain specification includes functional constraints or advice for the selection of resources. This information acts as control knowledge that selects, prefers or rejects particular alternative instantiations. The constraints can—and in many cases do—refer to the world state. For example, in a transportation environment, a sensible control rule would prefer to load the trucks that are at the same location as the packages to move.

New subgoals: When a new action is introduced into a plan, it typically also introduces a set of new subgoals. To some extent, the question of what subgoals are introduced is fixed: the planner must make sure that each of the preconditions of the new action is achieved. However, there are still two ways in which the members of the set of "what must be achieved" will depend upon the current state of the world:

- A reasonable thing for planners to do is to assume that preconditions that are currently true in the world do not need to be planned for. In Prodigy this happens automatically as a result of means-end analysis; in POCL planners it is achieved by a preference for re-use of (i.e., link to) the initial step.
- Planning systems which can handle universally quantified preconditions expand them into finite conjunctions involving all the literals that currently satisfy the quantified literal. For instance, given a precondition that all the packages in Pittsburgh be loaded onto a truck, the planner would determine which packages are known to be in Pittsburgh, and would create a precondition of getting each such package loaded onto a truck.

The current world state may influence planning decisions in yet more ways: for instance, when steps in the plan have conditional effects, the planner may infer that these effects will become true if their triggering conditions are currently true. For the current paper, however, we will focus on the influences outlined above.

Plan-Based and Alternative-Based Monitors

The generation of monitors follows directly from the use of the world state in making plan decisions, as just described. The monitors established during planning fall into two broad classes:

Plan-based monitors: At each point in the planning process, there is a current best plan (P) under consideration. Plan-based monitors represent world-state features that directly influence P. This includes, for instance, preconditions of all the operators in P. Some of these will be true when they are added to P; they therefore must be monitored, because, should they become false, P will fail unless additional planning is performed. Other preconditions will be initially false; should they become true, then the portions of P that established them may become unnecessary. These types of dependencies have been recognized in the work on execution monitoring as far back as the development of triangle tables (Fikes, Hart, & Nilsson 1972). We are performing this monitoring during the planning process—not just during execution—and are also considering a wider set of monitors and using them for a wider set of transformations.

Alternative-based monitors: We are particularly interested in planning situations in which the planner has the potential to form multiple alternative plans to achieve its objectives. A novel aspect of our approach is that we not only monitor features of the world that affect the current plan, but also features of the world that play a role in the decision to select that plan over alternative possibilities. For instance, it may be important to monitor the preconditions of some operator that was *not* selected: should they become true, then it may pay for the planner to "change its mind" and use that operator after all.

Plan-based and alternative-based monitors are clearly related. Every time the planner needs to make a decision among alternatives, it applies its evaluation function and selects a particular candidate plan. The selected plan gives rise to the plan-based monitors. At the same time, the alternatives considered give rise to alternative-based monitors. As the world state is dynamically changing, the planner remembers alternatives that it judged less valuable, monitoring the world state to see if that judgement should be changed.

Monitor Types

Within each class of monitors, we can identify several monitor types; so far, we make use of three distinct types:

Subgoal monitors: These encode all the preconditions and bindings of operators that have been considered in the planning process so far, including the operators in the current plan. The status of preconditions and their bindings can provide rationales for action selection, step instantiation, and new subgoal decisions, as explained above. In general, the plan-based subgoal monitors will be more fully instantiated than the alternative-based ones.

Usability-condition monitors: These represent the usability conditions of operators. Plan-based usability-condition monitors are important, because if one of them becomes false, the plan will need to be revised, with the operator in question being replaced. (Imagine that the current plan involves landing a helicopter in Pittsburgh, and Pittsburgh becomes fogged in.) Alternative-based usability-condition monitors play a different role. If one of them represents a currently false condition, then should that condition become true, the planner may re-consider using the operator that was initially unusable. (Imagine that the plan to land in Washington, DC was preferred, but Washington did not have clear weather when the initial planning decision was made.)

Quantified-condition monitors. A third type of monitor is generated by universally quantified preconditions of operators considered during the planning process. Given a precondition with a universally quantified predicate P, a monitor will be established to track the dynamics of the set denoted by P. (If the precondition involves loading all the packages currently in Pittsburgh, the world will be sensed to notice when the number of packages in Pittsburgh changes.) Plan-based quantified-condition monitors help ensure plan correctness: e.g., if a new package is delivered to Pittsburgh, the plan must be expanded to ensure that it is loaded as well. Alternative-based quantified-condition monitors suggest possible changes in planning decisions. (If the precondition for some alternative involves moving all the trucks now on the loading dock, then if the number of trucks on the loading dock goes to zero, the attractiveness of that alternative may increase.)

Plan Transformations

Whenever a monitor detects that a potentially relevant change in the world occurred, the planner may decide to attend that change, and then transform the plan. We organize *plan transformations* into three different categories:

Add to plan: Sometimes a change in the world necessitates extending the current plan. Two types of monitors can lead to this type of transformation. First, a plan-based subgoal monitor may fire when a condition C that was believed true has become false. Because C was believed true, the planner will have linked it to the world state, and not considered actions to achieve it. If it has now become false, then (assuming the current plan is maintained), C must be added to the set of open conditions. Second, a plan-based quantified-condition monitor may fire, representing a change in the extension of predicate. For example, suppose the quantified-condition monitor is keeping tracking of the set of all packages at the Pittsburgh depot. If a new member of that set is discovered, then the precondition that gave rise to that monitor may no longer be satisfied: if all the packages at the Pittsburgh depot must be loaded into the truck, then the goal of loading the newly discovered object must be added to the planner's open conditions.

Cut from plan: This transformation occurs in exactly the opposite situations from those described just above. When a portion of the current plan serves to establish some condition C, it may become possible to cut it out, should C become true.[2] Cuts may result from the firing of plan-based subgoal monitors and plan-based quantified-condition monitors: the latter lead to cuts when the satisfied set is reduced.

Jump in plan: Where the previous two transformations result only from plan-based monitors, and can be viewed as making improvements to the plan currently under consideration, the third may result from any type of monitor, and can best be viewed revisiting an earlier planning decision. Specifically, whenever any type of alternative-based monitor fires, it may indicate that a previously rejected alternative has now become more attractive (say, because some of its subgoals have now become true). The planner may then perform a "jump," i.e., change its focus of attention to the alternative. Note that plan-based monitors can also suggest jumps. Plan-based subgoal and universal-condition monitors can indicate that the current plan is not as attractive as it was before (say, because some of its subgoals, which had been true, have now become false). Plan-based usability-condition monitors always suggest jumps,

[2]The planner needs to determine that the affected portion of the plan does not also play some other role.

Monitor Class	Monitor Type	World State Change	Suggested Transformations
Plan-Based	Subgoal	T → F	{add,jump}
		F → T	{cut}
	Usability-condition	T → F	{jump}
	Quantified-condition	increased extension	{add,jump}
		decreased extension	{cut}
Alternative-Based	Subgoal	F → T	{jump}
	Usability-condition	F → T	{jump}
	Quantified-condition	decreased extension	{jump}

Table 1: Monitor Generation.

since their firing indicates that an operator in the plan is now longer usable: an alternative must be sought.

These transformations capture the core of the changes that occur when planning and sensing are interleaved. We have been also investigating refinements of these transformations, in particular in terms of a rich taxonomy of goal transformations (Cox & Veloso 1998).

During plan generation, each monitor can be tagged with the plan transformation(s) it may suggest. When a monitor fires, deliberation can then determine whether any of the suggested transformation should be performed. Table 1 summarizes our discussion and shows the transformations associated with each type of monitor.

Implementation

We have implemented planning with rationale-based monitors within the Prodigy4.0 planner (Veloso *et al.* 1995). Table 2 sketches the overall algorithm.

Prodigy4.0 is a state-space nonlinear planner. On its primary cycle it may do one of two things. It may select a pending goal, i.e., one that is not satisfied in the current state, and update the plan by adding a step that achieves that goal by adding it to its *tail plan* (Step 3 in the algorithm), or it may select an applicable step, i.e., one for which all of the preconditions are currently satisfied, and add it to the end of a totally ordered *head plan* (Step 4 in the algorithm). In the latter case, the state changes that result from the applied action are performed and produce the new planning state.

To incorporate rationale-based monitoring, two primary changes to the algorithm are needed, as shown in boldface in Table 2. First, rationale-based monitors are generated whenever the plan has been updated. Second, sensing is performed to check the status of the world conditions being monitored, and plan transformations are performed in response.

1. Terminate if the goal statement is satisfied in the current state.
2. Compute the set of *pending goals* $\mathcal{G}$, and the set of *applicable operators* $\mathcal{A}$. A goal is pending if it is a precondition, not satisfied in the current state, of an operator currently in the plan. An operator is applicable when all its preconditions are satisfied in the state.
3. Either
   - Choose a goal G from $\mathcal{G}$
   - *Expand* G, i.e., get the set $\mathcal{O}$ of *relevant instantiated operators* that could achieve the goal G,
   - Perform action selection.
   - Perform step instantiation.
   - Add new step to tail plan.
   - **Generate new monitors.**
4. or
   - Choose an operator A from $\mathcal{A}$.
   - *Apply* A: Add A to the head plan and get new current state.
5. **Sense for fired monitors, and perform planning transformations.**
6. Go to step 1.

Table 2: A skeleton of Prodigy4.0's planning algorithm with rationale-based monitoring.

Signal-Handling Interrupts for Sensing

To incorporate sensing during planning in Prodigy, we have relied on its interrupt-handling mechanism (Stone & Veloso 1996). This mechanism is a useful part of the Prodigy system, and has previously been used in a number of ways.[3] In our current implementation, changes to the world state may be made during the planning process; at the end of each planning cycle, the planning process is interrupted and updates to the environment are "sensed" and incorporated into the planner's state. We have also implemented a generalization, in which sensing occurs only every n cycles, where n is a user-specified parameter.

Currently, world-state changes can only be specified as literals. Each literal is tagged as being satisfied in

[3]For example, Prodigy's GUI relies on the interrupt-handler to communicate user commands and display results after each planning cycle.(Cox & Veloso 1997)

the state or not. Multiple changes can be observed during a single sensing operation.

Monitor Generation in Prodigy

New monitors may be generated whenever Prodigy updates the tail plan. A pending goal is selected, and an operator is added to the plan to achieve it. The goal unification with the effects of the operator provides some initial bindings for the preconditions. The remaining variables of the preconditions are then bound, and new subgoals are added. Monitors are then created to track the world state features that led to each of these decisions.

The monitors are implemented as Prodigy signal-handling functions. Thus, if a monitor is established for some condition P, then the introduction of the literal P into the world state will cause that monitor to fire at the end of the cycle following its introduction. Table 3 provides the pseudo-code for our implemented monitor generation algorithm.

- Let O be a selected or alternative instantiated step and let B be the corresponding binding search node.
- Let pre(O) denote the regular preconditions of O.
- Let q-expr(O) denote the quantified preconditions of O and let each precondition in q-expr(O) be of the form $p(x), \forall x, s.t. f(x)$.
- Let $\mathcal{U}$, $\mathcal{Q}$, and $\mathcal{S}$ respectively denote the sets of Usability, Quantified-condition, and Subgoal monitors.
- Let usability-condition-p be true for the usability conditions.

```
for each precond $p$ in pre($O$)
  if usability-condition-p(p)
      then $\mathcal{U} = \mathcal{U} \cup$ usability-monitor($B, p$)
      else $\mathcal{S} = \mathcal{S} \cup$ subgoal-monitor($B, p$)
for each precond $p(x), \forall x, s.t. f(x)$ in q-expr($O$)
  $\mathcal{Q} = \mathcal{Q} \cup$ quantified-condition-monitor($B, f$)
  for each expansion $p(x_i)$ s.t. $f(x_i)$ is true
      $\mathcal{S} = \mathcal{S} \cup$ subgoal-monitor($B, p(x_i)$)
```

Table 3: Algorithm to generate monitors in Prodigy4.0.

Plan Transformations

When a monitor fires, it suggests a possible plan transformation. We implemented most of the plans transformations described above. The implementation did not requires changes to the Prodigy architecture. This is because Prodigy allows control of the planning search to be manipulated through declarative structures, called control rules (Veloso *et al.* 1995; Borrajo & Veloso 1996). Control rules enable heuristic redirection of the search for operators to achieve goals, for bindings to instantiate operators, and for the next node to be expanded. The default is to perform depth-limited search through planning decisions. But given a control rule that selects a specific search node, this default can easily be overturned.

Plan transformations are implemented as control rules that lead to movement through the search tree. The implementation within Prodigy depends upon its distinction between the tail plan and the head plan, introduced in the discussion of Table 2.

Prodigy adds new steps to the tail plan and recursively plans for the preconditions of those steps. When a step in the tail plan is found to have all of its preconditions true in the state, that step can be *applied* (see Step 4 in Table 2), i.e., moved to the head plan, and a new state is computed.

Our implementation of plan transformation needs to take care of two situations:

- Consider that the change detected refers to a precondition of a step in the tail plan. Then
  - If this precondition was already planned for, then the planner cuts that unneeded plan.
  - If this precondition has not been planned for, then the planner automatically incorporates it into its open preconditions through the computation of the pending goals.
- Consider that the change detected refers to a precondition of a step already in the head plan. Then the planner backtracks to the node previous to the application of the step, to do further planning.

Similarly, when an alternative-based monitor fires, the planner selects the appropriate alternative node from which to pursue the plan generation. Control rules are generated dynamically to select the appropriate change of focus in the search.

For example, Table 4 shows a control rule that is associated with a jump transformation and automatically generated when a monitor fires. The rule is executed only through the use of a "one-shot" function (it returns t upon the first invocation and nil otherwise). If the binding node (bnode) is still expandable, it will select that node as the node to pursue planning from.

bode *is the specific node associated with the firing monitor.*

```
(eval '(control-rule
        ,(gentemp "JUMP-2-READY-STEP-")
       (if (and (one-shot ,done-var)
                (candidate-node ,bnode)))
       (then select node ,bnode))))
```

Table 4: Dynamically created control rule to perform a jump transformation.

Experiments

Using the implementation described above, we designed and performed a preliminary set of controlled experiments to illustrate aspects of our approach.

Alternative-based subgoal monitors. We created an artificial domain to test the generation of subgoal monitors and the effect of the alternative-based ones. The goal of this experiment was to illustrate the implementation showing that the basic idea is feasible, and, more specifically, that the use of alternative-based monitors can lead to better plans, compared to those that would be formed if no monitoring occurred. The cost of sensing in our simple experiments was not a significant overhead, although we recognize that it may become more of a factor if a large set of monitors is generated. We thus are currently developing techniques to prioritize rationale-based monitors.

In this experiment, we created an artificial domain. At a certain point in the planning process, we changed the world state, and made true a condition that would suggest a jump to a different partial plan.

The operators in our artificial domain are shown in Figure 1. The operators fall into three classes. First, there are n operators $O_1(x) \ldots O_n(x)$, where each $O_i(x)$ has two preconditions, g_{i+1} and $a(x)$ and a single effect g_i. (Note that x ranges over a set $B_k = \{x_1, \ldots, x_k\}$ of k possible bindings. Therefore there are k many instantiated operators to achieve each goal g_i.) Second, operator $O_{n+1}()$ has no preconditions and the effect g_{n+1}. Finally, operator $O_*(x)$ has no preconditions and the effect $a(x)$.[4]

operator $O_i(x)$	operator $O_{n+1}()$	operator $O_*(x)$
$i = 1, ..., n; x \in B_k$	pre: ()	pre: ()
pre: $a(x)$ & g_{i+1}	add: g_{n+1}	add: $a(x)$
add: g_i	del: ()	del: ()
del: ()		

Figure 1: Experimental domain for alternative-based monitors.

In each planning problem, we set the initial state to be one in which only $a(x_1)$ is true, and set the goal to be g_1. By varying n, we can vary the complexity and length of the solution, and by varying k, we vary the number of possible bindings for x and therefore the branching factor and the number of alternatives available. As an example, consider $n = 3$ and $k = 5$. As $a(x_1)$ is the only true a literal in the initial state, a solution plan is $O_4(x_1)$, $O_3(x_1)$, $O_2(x_1)$, $O_1(x_1)$. If, however, for example both g_2 and $a(x_5)$ become true during the planning process, then $O_1(x_5)$ becomes an alternative shorter solution. The purpose of monitoring is to observe such a change, and suggest a jump to the shorter plan.

In the experiment reported here, we varied n, the number of steps in a solution plan if the world state does not change; n ranged from 1 to 30. We fixed k, the number of binding possibilities for the planning operators, to 2. During planning, two monitors fire making true the literals g_2 and $a(x_2)$. We vary the time at which these two monitors fire during the planning process, namely after 0, 10, and 20 planning steps. The results are shown in Figure 2, which plots the total planning time as a function n.

[4]Note that $a(x)$ is not a usability condition, because O_* can achieve it.

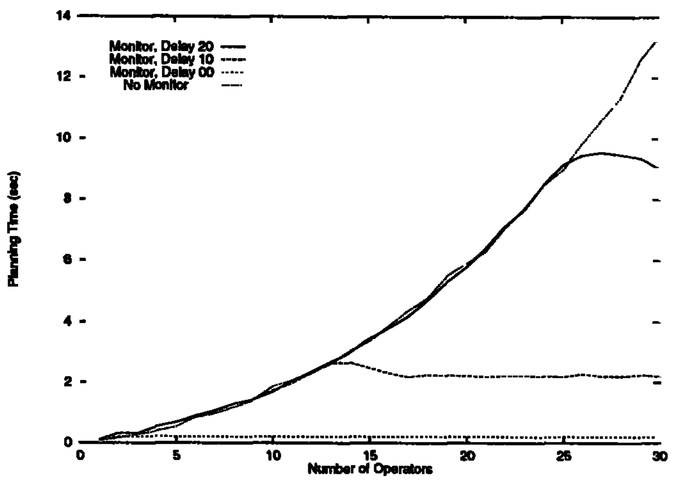

Figure 2: Alternative-based subgoal monitors: Planning performance with and without rationale-based monitoring. The curves refer to different delays of the firing of the monitors during the planning process.

As can be seen, when the environment does not change, the amount of time to generate a solution increases with n. However, with the rationale-based monitors, the planner can react to changing circumstances and find a plan more quickly. As would be expected, when the changes occur later (say, after 20 planning cycles rather than 10), the savings to the planner is reduced, because it has already performed significant planning.

Alternative-based usability monitors. Generating and responding to rationale-based monitors can allow a planner to generate non-faulty plans. To show the effect of usability monitors, we performed a similar experiment to the one above. By simply removing the operator O_* in the domain shown in Figure 1, $a(x)$ becomes a usability condition. In the initial state, all (or a set) of $a(x_k)$ are satisfied in the initial state. For example for $k = 2$, both $a(x_1)$ and $a(x_2)$ are satisfied. In this new experiment, however, the usability monitor is the only one that fires, detecting the deletion of $a(x_1)$ from the state. As a is now an unachievable predicate, the planner needs to jump to an alternative where its usability condition is true. The final plans generated are of length n in all situations, independent of the alternative taken. However, if no monitoring is done and the world changed, the plans generated are faulty. Our experiments showed that the planner with rationale-based monitoring produces the correct plan, accounting therefore for the state dynamics while planning.

Plan-based quantified-condition monitors. Another interesting example is the use of quantified-condition monitors, that once again allows for a planner to avoid generating a faulty plan. We performed experiments in several domains, in particular in a realistic simple military domain.

Consider the previously introduced general formulation of a universally-quantified precondition, $p(x), \forall x, s.t. f(x)$. When our quantified-condition monitors detect an additional value of x, e.g. x_i, for which $f(x_i)$ becomes true in the world, the planner adds a new universal precondition to achieve $p(x_i)$. Without the monitors and without $p(x_i)$ being true in the initial state, no step in the plan would be generated to achieve $p(x_i)$. The plan would then be faulty. With rationale-based monitors that use the universally quantified expression in the operator definition to watch for this event, the planner can react to the new information to create a correct plan.

We have implemented a number of examples of quantified condition monitors in the military domain of air campaign planning. For instance, in our domain we can have a top-level goal to make a particular river impassable. The operator to achieve this goal requires that all of the river crossings (e.g., bridges) such that the crossing enables troop movements, be destroyed. The planner posts subgoals to destroy all of the crossings known in its state. A quantified-condition monitor is then created to watch for additional crossings that enable such movement. During planning, when our monitor triggers upon the detection of a new river crossing, then the planner correctly adds a new subgoal to destroy the new crossing discovered. Similarly, if a crossing is disabled, the planner correctly cuts any planning that was done for that crossing.

Discussion and Conclusion

In this paper, we have presented our work on developing a framework for fully integrated planning and execution in dynamic environments. In this initial phase of this work, we have concentrated on the problem of enabling a planning system to deal with changes in the environment. A central concern has been with the question of how to focus the sensing performed by the system, so that it responds appropriately to relevant changes, but is not overly taxed by attempting to monitor all the changes that could possibly occur in the world. We observed that planning is essentially a decision process, and that many of the decisions made by planners depend upon the current world state. Therefore, one way to help achieve the right degree of sensitivity to the world state is to have the planner focus on those changes that have influenced its planning decisions so far.

We therefore developed the idea of rationale-based monitors, which including both the selections made in the plan under development and the alternatives considered. Rationale-based monitors encode features of the world that the current plan depends on, as well as features of the world that led to the rejection of alternatives. When a rationale-based monitor fires, it can therefore suggest that the current plan be modified (i.e., by adding new elements to it or cutting portions of it), or it can suggest that the planner revisit one of its previous planning decisions, for instance, replacing one of the operators used to achieve some goal.

We have implemented an initial version of rationale-based monitoring within the Prodigy planner. We have done controlled experiments in several domains, including military and artificial domains. The results show that planning with the rationale-based monitors can reduce the total planning and increase the correctness of the plans generated.

With the monitor-based framework in hand, we can now address a number of remaining, important questions. The one we find particularly critical involves the question of meta-level deliberation. The firing of a rationale-based monitor suggests an opportunity for improving the plan being developed. Reasoning must be performed to decide whether and/or which transformations should be applied. For example, alternative-based monitors suggest that a previously rejected alternative *might* be re-considered, but it is not necessarily always the case that they should be. When a plan-based subgoal monitor fires, it suggests that either additional open goals need to be added or a different operator should be tried. We are thus currently focusing on the development of deliberation strategies for handling monitors.

Acknowledgements This research is sponsored in part by the DARPA/AFRL under grant number F30602-97-2-0250 for the first and third authors, and by the Air Force Office of Scientific Research contract number F49620-96-1-0403 and National Science Foundation grants IRI-9258392 and IRI-9619579 for the second author. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies or endorsements of the U. S. Government.

References

Blythe, J., and Veloso, M. M. 1992. An analysis of search techniques for a totally-ordered nonlinear planner. In *Proceedings of the First International Conference on AI Planning Systems*, 13–19.

Blythe, J. 1998. *Planning under uncertainty in dynamic domains.* Ph.D. Dissertation, Computer Science Department, Carnegie Mellon University.

Borrajo, D., and Veloso, M. 1996. Lazy incremental learning of control knowledge for efficiently obtaining quality plans. *AI Review Journal. Special Issue on Lazy Learning* 10:1–34.

Boutilier, C.; Dean, T.; and Hanks, S. 1995. Planning under uncertainty: structural assumptions and computational leverage. In *Proceedings of the European Workshop on Planning.*

Bratman, M. E.; Israel, D. J.; and Pollack, M. E. 1988. Plans and resource-bounded practical reasoning. *Computational Intelligence* 4:349–355.

Collins, G., and Pryor, L. 1995. Planning under uncertainty: some key issues. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence.*

Cox, M. T., and Veloso, M. M. 1997. Supporting combined human and machine planning: An interface for planning by analogical reasoning. In *Case-Based Reasoning Research and Development, Proceedings of ICCBR-97, the Second International Conference on Case-Based Reasoning,* 531–540. Springer Verlag.

Cox, M., and Veloso, M. 1998. Goal transformations in continuous planning. In *Forthcoming.*

Draper, D.; Hanks, S.; and Weld, D. 1994. Probabilistic planning with information gathering. In *Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems*, 31–36.

Drummond, M., and Bresina, J. 1990. Anytime synthetic projection: Maximizing the probability of goal satisfaction. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 138–144. Boston, MA: AAAI Press/The MIT Press.

Fikes, R. E.; Hart, P. E.; and Nilsson, N. J. 1972. Learning and executing generalized robot plans. *Artificial Intelligence* 3:251–288.

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

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

Georgeff, M. P., and Ingrand, F. F. 1989. Decision-making in an embedded reasoning system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 972–978.

Goldman, R., and Boddy, M. 1994. Epsion-safe planning. In *Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence.*

Haddawy, P., and Suwandi, M. 1994. Decision-theoretic refinement planning using inheritance abstraction. In *Proceedings of the Second International Conference on AI Planning Systems*, 266–271.

Kushmerick, N.; Hanks, S.; and Weld, D. 1995. An algorithm for probabilistic planning. *Artificial Intelligence* 76:239–286.

Onder, N., and Pollack, M. E. 1997. Contingency selection in plan generation. In *Proceedings of the 4th European Conference on Planning.*

Penberthy, J. S., and Weld, D. S. 1992. UCPOP:A sound, complete, partial order planner for ADL. In *Proceedings of KR-92*, 103–114.

Peot, M., and Smith, D. E. 1992. Conditional nonlinear planning. In *Proceedings of the First International Conference on AI Planning Systems (AIPS-92)*, 189–197.

Stone, P., and Veloso, M. M. 1996. User-guided interleaving of planning and execution. In *New Directions in AI Planning.* IOS Press. 103–112.

Veloso, M. M.; Carbonell, J.; Pérez, M. A.; Borrajo, D.; Fink, E.; and Blythe, J. 1995. Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical Artificial Intelligence* 7(1):81–120.

Williamson, M., and Hanks, S. 1996. Flaw selection strategies for value-directed planning. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems*, 237–244.