Abstract

Traditional approaches to dealing with uncertainty in planning have focused on finding plans that prevent all potential failures. Though such plans are robust, their creation is computationally expensive. This model of planning does not capture the facts that 1) many times the most likely branch of execution succeeds and 2) even when that branch fails, replanning during execution frequently provides an alternate path to the goal. In reality, the only failures that need to be planned for before execution are those that are unrecoverable, thereby preventing achievement of the goals. We have developed a framework called Precautionary Planning that combines interleaved planning and execution with limited contingency planning. Precautionary Planning adopts the view that contingency planning should be a last resort and is not desirable when replanning is possible. In this framework, a robust initial plan is generated using a fast deterministic planner. Next, the plan is analyzed to find potential points of failure, which are identified as recoverable or unrecoverable. Recoverable failures are left in the plan and are repaired through replanning at execution time. For each unrecoverable failure, an attempt is made to improve the chances of recovery, by adding “precautionary” steps such as taking along extra supplies or tools that would allow recovery if the failure occurs.

Introduction

Uncertainty is pervasive in many of the planning problems relevant to NASA. For example, in Mars Rover operations, there is inherent uncertainty about such things as the duration of tasks, the power required, the data storage necessary, and environmental factors that influence things like battery charging, or which scientific tasks are possible or important.

Traditional approaches to probabilistic planning under full observability involve the construction of a policy using some form of value or policy iteration (Puterman 1994; Boutilier, Dean, & Hanks 1999), or a heuristically guided forward state space search such as LAO* (Hansen & Zilberstein 2001) or LRTDP (Bonet & Geffner 2003). One assumption that is shared by most planners implementing these approaches is that all possible failures in the plan should be accounted for at planning time. In other words, the policies are complete contingency plans that account for every possible outcome that might occur given the action descriptions. These policies are certainly robust plans, but it can be computationally expensive to generate them, since the search space explodes as the number of relevant action outcomes increases.

This work is inspired by the somewhat surprising success of FF-rePlan in the first two International Probabilistic Planning Competitions held in 2004 (Younes et al. 2005) and 2006 (Bonet & Givan 2006). In 2004, FF-rePlan won the competition, and though it was not submitted to the 2006 competition, the organizers ran it for comparisons. In both competitions, FF-rePlan was able to cover more problems faster than the other planners. In place of dealing with uncertainty at planning time FF-rePlan plans optimistically, considering only the most likely outcome of each action, then monitors execution and replans when failure occurs. However, replanning is useless when an unexpected action outcome causes unrecoverable failure. For example, consider a problem that requires driving through a remote stretch of territory and then crossing the border into a different country. A tire can go flat during the trip and can only be replaced if there is a spare tire in the vehicle. Additionally, gas stations may be far apart and may be closed. The most optimistic plan is to just start the journey, assuming that everything will work out. This is exactly the plan that FF-rePlan would generate and follow. Unrecoverable failure would occur if a tire went flat and there was not a spare, or if fuel ran out because a gas station was closed. In many cases, though, outcomes like this are only unrecoverable when left until discovered at execution time. In this example, planning ahead by putting a spare tire in the vehicle and bringing along a container of gas would prevent these dead-ends.

In this paper we describe an interleaved planning and execution framework called Precautionary Planning that takes advantage of the speed of replanning, but considers the possibility of unrecoverable failures and attempts to avoid them. In this framework, first a deterministic planner is used to generate a plan that
has a high probability of success. Then, a look-ahead is performed to find action outcomes that would result in unrecoverable failures. At such points, an attempt is made to improve the plan so that either the undesirable outcome will not occur or it is possible to recover from the outcome if it does occur. One can think of this as a form of limited or incremental contingency planning (Dearden et al. 2002; 2003; Meuleau & Smith 2003) since this approach starts with an unconditional seed plan and attempts to incrementally improve that plan by considering only the most problematic action outcomes.

In the following sections we describe the details of Precautionary Planning including the interleaved planning and execution framework, the generation of an unconditional seed plan, the identification of unrecoverable failures, and the repair of those failures. Finally, we discuss a preliminary implementation, related work and some future directions.

**Precautionary Planning**

We assume that we are given a probabilistic planning problem represented in PPDDL 1.0 (Younes et al. 2005) where action outcomes are fully observable. Figure 1 shows a simplified version of the problem described in the introduction expressed in PPDDL 1.0. For purposes of this paper we do not consider either action costs or rewards, so the objective is simply to find a plan that has maximum probability of success.

Figure 2 and Algorithm 1 show a sketch of the top level algorithm for Precautionary Planning. First, a deterministic planner is used to generate a seed plan that reaches the goal with high probability of success. Next, an analysis and repair cycle is performed to improve the net probability of the plan. The analysis searches for an outcome that is both sufficiently probable and results in a dead end (the goal cannot be achieved from this outcome). If possible, this outcome is repaired by adding actions to the plan to either: 1) avoid the problematic outcome, 2) allow the outcome to be repaired, or 3) improve overall probability of reaching the goal so that the outcome becomes less important. The analysis and repair cycle continues on the improved plan until plan probability is sufficiently high, no more improvement is possible, or a time limit is exceeded. At this point, the next action in the plan is executed. If execution results in an unplanned outcome, a new plan must be generated to reach the goal from this unexpected state. Analysis and repair is then performed on this new plan before execution of its first step. Alternatively, if execution results in an expected outcome, the precautionary planner can continue on with the remainder of the existing plan. However, even in this case, we perform analysis and repair of the remainder of the plan before executing another step. We do this because 1) the probability of subsequent outcomes in the remainder of the plan may have changed (increased) due to the outcome of the step just executed, and 2) the previous analysis and repair cycle may have been limited

### Domain description

```plaintext
(define (domain treacherous-drive)
  ...  
  (:action get-tire
    :precondition (at-start)
    :effect (have-tire))

  (:action get-passport
    :precondition (at-start)
    :effect (have-passport))

  (:action drive-from-start
    :precondition (at-start)
    :effect (and (not (at-start))
               (probabilistic 3/5 (at-end)
                            2/5 (and (flat-tire)
                                    (along-route)))))

  (:action replace-tire
    :precondition (flat-tire)
    :effect (not (flat-tire) (tire-replaced))

  (:action drive-from-along-route
    :precondition (and (along-route)
                       (tire-replaced))
    :effect (and (not (along-route)) (at-end))

  (:action cross-border
    :precondition (and (at-end)(have-passport))
    :effect (border-crosssed))
)
```

### Problem description

```plaintext
(define (problem drive-problem)
  (:domain treacherous-drive)
  (:init (at-start))
  (:goal (border-crossed)))
```

Figure 1: A probabilistic domain and problem expressed in PPDDL1.0.

by time.

### Seed Plan Generation

Though the domains we work with are probabilistic, our framework uses a fast, deterministic planner to generate the seed plan. This requires conversion of the probabilistic domain to a deterministic domain. Since a seed plan with high probability of success is desired, preference must be given to high probability outcomes in the conversion. Algorithm 2 shows the method of converting a domain and problem written in PPDDL1.0
Algorithm 1
planning/execution algorithm

PRECAUTIONARY PLANNER(problem)
1: PDDL-problem ← CNVRT-TO-DETERM(problem)
2: plan ← DETERM-PLANNER(PDDL-problem)
3: repeat
4: REPAIR-FAILURES(plan, PDDL-problem)
5: cur-state ← EXECUTE-STEP(plan)
6: if unplanned outcome then
7: init-cond(PDDL-problem) ← cur-state
8: plan ← DETERM-PLANNER(PDDL-problem)
9: end if
10: until plan is null

(Younes et al. 2005) to PDDL2.2¹ (Edelkamp & Hoffman 2004).

Algorithm 2

function CNVRT-TO-DETERM(problem)

Part A: Convert domain
1: PDDL-domain ← determ parts of DO-
2: for all prob actions Aᵢ in DOMAIN(problem) do
3: total_prob ← 0
4: for all prob outcomes i of A do
5: total_prob ← total_prob + pr(i)
6: Aᵢ ← determ copy A with only outcome i
7: SET-COST(Aᵢ, -1 * log(pr(i))
8: if i is conditional then
9: ADD-PRECOND(Aᵢ, condition of i)
10: end if
11: ADD-ACTION(PDDL-domain, Aᵢ)
12: end for
13: if total_prob < 1 then
14: Aᵢ ← determ copy A when no prob outcome
15: SET-COST(Aᵢ, -1 * log(1 - total_prob))
16: ADD-ACTION(PDDL-domain, Aᵢ)
17: end if
18: end for

Part B: Convert problem
1: PDDL-problem ← PROBLEM(problem)
2: SET-INITIAL-COST(PDDL-problem, 0)
3: SET-METRIC(PDDL-problem, minimize cost)

To convert the domain, we use an approach similar
to that of Jiménez, Coles, & Smith (2006); each prob-
abilistic action is broken into several deterministic ac-
tions, one for each probabilistic outcome (loop starting
in part A, line 2). To force the deterministic planner
find a high probability plan, the plan metric fea-
ture of PDDL2.2 is used. For each new action that is
created, the probability of its outcome is converted to
an additive cost equal to the negative logarithm of the
probability (part A, lines 7, 15). More precisely, if A
is a probabilistic action with outcomes O₁, . . . , Oₖ and
probabilities P₁, . . . , Pₖ, we create deterministic actions
A₁, . . . , Aₖ each having the same conditions as A, effects
of O₁, . . . , Oₖ respectively, and costs Cᵢ = − log Pᵢ. The
problem is then converted by initializing cost to 0 and
setting the plan metric as minimize cost (part B, lines
2, 3). Figure 3 shows the converted form of the action
drive-from-start and the converted problem for the
domain and problem in Figure 1.

\[
\begin{align*}
\text{Converted action } & \text{drive-from-start} \\
& (:action \text{ drive-from-start-1}) \\
& :precondition (\text{at-start}) \\
& :effect (\text{and (not (at-start) (at-end))}) \\
& (\text{increase (cost) 0.222})) \\
& (:action \text{ drive-from-start-2}) \\
& :precondition (\text{at-start}) \\
& :effect (\text{and (not (at-start) (along-route))}) \\
& (\text{increase (cost) 0.397}))
\end{align*}
\]

Figure 3: Parts of the probabilistic domain and problem
converted to PDDL2.2.

Recognizing Unrecoverable Outcomes
Once the seed plan is generated, actions that could
cause unrecoverable failure must be recognized. Given
an action in the plan, it must first be mapped back
to its counterpart in the probabilistic domain to find
other possible outcomes. A determination must then
be made as to whether any of the alternate outcomes
could cause unrecoverable failure. A simple way to test
for this is to change the initial conditions of the problem
to represent the state of the world when an alternate
outcome occurs and then call the deterministic planner.
If a plan cannot be found, the failure is unrecoverable.
If a plan does exist, it is probabilistic, like the seed plan,
and so must also be analyzed for unrecoverable failures.
This approach accurately locates dead ends, but incurs
overhead by calling the planner repeatedly for each al-
ternative outcome in the plan. As a result, the expense
of this approach gets out of hand as the number of un-
certain outcomes in the plan increases. A potentially
faster alternative is to construct a plan graph and use
it for reachability analysis. By propagating probabil-
ity estimates through the plan graph (Bryce & Smith
2006), we can estimate the probability of reaching the
goal after the alternate outcome occurs. The disad-
vantange of this approach is that plan graph reachability
is optimistic and could lead us to believe that an outcome
is recoverable when it is not.
Alternatively, in some cases it may be possible to do domain analysis and prove in advance that certain action outcomes are reversible – that is, a sequence of actions exists that can be executed to return the world to the state before the action in question was executed. In such cases, the outcome is recoverable, and it is not be necessary to run the planner to determine this. Once the world is returned to the previous state, the action can be executed again. If the desired outcome occurs, the plan can continue. Otherwise the reversal steps can be repeated as long as the alternate outcome occurs.

Ultimately, we believe that a combination of these three techniques will be required to efficiently assess action outcomes. First, one should check an outcome to see if it is reversible. If not, a plan graph could be used to see if the outcome is unrecoverable or has low probability of recovery. Finally one could invoke the planner on high probability outcomes not identified as either reversible or unrecoverable by the previous two techniques. This limits the expense of the planner to those outcomes that are relatively important and still suspect.

Repairing Unrecoverable Outcomes

Once an action with an unrecoverable outcome is identified, we want to repair the plan so that the overall probability of success is improved. There are potentially four different ways in which this can be accomplished: 1) avoid the problematic outcome through confrontation, 2) allow the outcome to be repaired by adding precautionary steps to the plan prior to the problematic action, 3) improve overall probability of reaching the goal by incorporating conformant actions, or 4) abandoning the plan and seeking another seed plan that is more repairable. This section discusses these four options in detail.

Confrontation

Sometimes the probabilistic outcomes of an action are contingent upon different conditions. For example, in the travel problem it might be that for the drive-from-start action, flat-tire can only occur when the tires are old. Confrontation of this condition would allow us to avoid flat-tire by always ensuring we have new tires before beginning drive-from-start. Let \( A_s \) be our action in the seed plan with unrecoverable outcome \( O_u \) and let \( C_u \) be the condition for that outcome. We want to modify the existing seed plan to force achievement of the negation of this condition, so that the bad outcome \( O_u \) can not occur. We can trick a deterministic planner into doing this as shown in Algorithm 3. We create a new version of this action \( A'_s \) which has all the original preconditions and effects of \( A_s \) (line 3) but also has the additional precondition \( \neg C_u \) (line 4) and a new unique effect (line 5). We force \( A'_s \) into the plan by adding this unique effect to the goal (line 6). We add \( A'_s \) to the domain (line 7) and solve the planning problem again. If a new plan is found, and the probability of this new plan is greater than that of the seed plan, the new plan replaces the seed plan, avoiding the unrecoverable outcome.

Algorithm 3

function Confrontation-Repair(plan, PDDL-problem)

1: \( A_s \leftarrow \) action in plan that can cause unrecoverable failure
2: \( C_u \leftarrow \) condition under which unrecoverable outcome of \( A \) occurs
3: \( A'_s \leftarrow \) copy of \( A_s \)
4: Add-Precondition\((A'_s, \neg C_u)\)
5: Add-Effect\((A'_s, \text{unique-effect})\)
6: Add-Goal\((PDDL-problem, \text{unique-effect})\)
7: Add-Action\((PDDL-problem, A'_s)\)
8: \( \text{plan} \leftarrow \) DETERM-PLANNER\((PDDL-problem)\)

Precautionary Steps

Another way to resolve an unrecoverable outcome is to insert precautionary actions into the plan that improve the probability of recovery. In the example problem, we can avoid unrecoverable failure when the tire goes flat by including a precautionary step of get-tire at the beginning of the plan. Recovery is then possible by executing fix-tire and drive-from-long-route when drive-from-start results in a flat tire.

As with confrontation, we can trick a deterministic planner into inserting precautionary actions into a plan. Algorithm 4 gives a sketch of the process. Let \( A \) be our action with unrecoverable outcome \( O_u \) and let \( A_s \) and \( A_u \) be the deterministic versions of \( A \) corresponding to the desired and unrecoverable outcomes of \( A \). The basic idea is to force the deterministic planner to find a plan that includes \( A_u \) but does not destroy anything that is needed to reach the goal when \( A \) has the desired outcome \( O_s \). To facilitate this, the seed plan is divided into three parts: the prefix (containing all actions preceding \( A_s \) plus the initial world state), \( A_s \), and the suffix (containing all actions following \( A_s \) plus the goal state). We create a new deterministic action \( A'_s \) that includes all the preconditions and effects of \( A_u \) (line 2). We also add a unique effect to \( A'_s \) that is then added to the goal to force \( A'_s \) into the plan (line 3, 4). The suffix of the seed plan \( \Pi \) is protected by identifying any conditions necessary for the suffix that are fulfilled in the prefix.

2There is a more efficient, but more complicated way of doing confrontation that avoids having the planner regenerate the plan suffix (those actions following \( A_s \) in the plan). Doing this requires that we analyze the causal structure of the plan suffix to find the set of conditions \( C_{suffix} \) that are satisfied by actions occurring in the plan prefix (those actions preceding \( A_s \) in the plan). If \( C_s \) are the preconditions of \( A_s \) we instead solve a new problem with goals \( G' = C_s \cup \neg C_u \cup C_{suffix} \). If a plan is found, it serves as a replacement for the prefix of the seed plan.
Algorithm 4

function Precautionary-Repair(plan, PDDL-problem)
1: A ← action in plan that can cause unrecoverable failure
2: \( A'_u \) ← copy of \( A_u \), the version of \( A \) with the unrecoverable outcome
3: Add-Effect(\( A'_u \), unique-effect)
4: Add-Goal(PDDL-problem, unique-effect)
5: for all preconditions \( pc \) in plan suffix supported by plan_prefix do
6: Add-Precondition(\( A'_u \), \( pc \))
7: end for
8: Add-Action(PDDL-problem, \( A'_u \))
9: new-plan ← DETERM-PLANNER(PDDL-problem)
10: branch ← plan_prefix
11: Add branch to new-plan after \( A'_u \)
12: plan ← new-plan

Figure 4: Action \( A'_u \) that is included in a plan to repair the unrecoverable outcome of a flat tire. The condition have-passport must be added as a precondition because it is needed by the suffix.

Figure 5 shows (a) the seed plan, (b) the new plan that is generated to repair the problem, and (c) the merged contingency plan for our example problem.

Conformant Solution

Sometimes it is possible to add steps to a plan that reach the goal through different means and thereby increase the overall probability of reaching the goal. In our example, suppose that the real objective was to deliver a document across the border. One way of increasing the chance of success would be to send a copy of the document in a separate vehicle. This is essentially a conformant rather than a contingent means of increasing the chance of success of a plan. As it turns out, conformant plans can result naturally when we apply

Algorithm 4. This is the case when the deterministic planner finds a plan to achieve the original goals, but the action \( A'_u \) (which was forced to be in the plan) is not on the critical path for any of those goals. In other words, the only purpose of \( A'_u \) is to achieve the unique effect that we added to the goal, and the resulting plan would still achieve the original goals if \( A'_u \) were removed from the plan. The reason this plan was not generated initially is because on its own it has a lower probability of success than the seed plan. However, inclusion of both the high probability seed plan and lower probability plan increases the overall probability of success. Since \( A'_u \) ensures the prefix of the seed plan can be executed, the two plans can be merged. More precisely, if \( \Pi' \) is the new plan, we can replace \( A'_u \) in \( \Pi' \) with \( A \) and add the suffix of the original seed plan to \( \Pi' \), but condition it on the successful outcome \( O_s \) of \( A \). This is basically the same as the plan merging step for Precautionary Planning. There is, however, one additional complication: some of the conformant planning steps might interfere with steps in the suffix of the seed plan. Thus the final step in the plan merging process is to condition any such steps on having an outcome for \( A \) other than \( O_s \). This process is similar to the conditioning process for threat resolution in POCL conditional planning described in Peot & Smith (1992).

Truly Unrecoverable Outcomes

This work is based on the observation that many failures are recoverable when planned for ahead of time. Though this is often the case, it is not always true. It is possible that adding the restrictions of Algorithms 3 and 4 to the planning problem prevents the generation of a plan. This, then, indicates a truly unrecoverable outcome. However, it does not necessarily mean that the problem is unsolvable. It may simply be that the seed plan should be abandoned and replaced with a new plan that has a lower probability of success on its own, but can be repaired to raise the overall probability to an acceptable level. To find the new plan, either the domain would have to be modified to prevent regeneration of the seed plan (this could involve removal of the action causing unrecoverable failure) or the deterministic planner would need to generate a succession of plans rather than a single plan. In either case, a new
seed plan should be considered. Of course it may ultimately turn out that no other seed plan is any better, and that the original seed plan with unrecoverable outcomes is the highest probability solution. As a result, this entire process should be considered as a search through a space of probabilistic plans at various stages of improvement.

**Searching for Unrecoverable Outcomes**

So far, we have assumed that, given a seed plan, all action outcomes are considered for repair. For larger problems this is generally not practical. Instead, we need to focus the improvement process on those outcomes that are likely to make the most difference in improving plan probability. There are several possible strategies for doing this, but we will focus here on a progressive greedy approach. This approach scans the plan starting at the beginning and progresses forward in time. For each action $A$ with an alternative (unplanned) outcome $O$ we consider the probability that $O$ will actually occur. This is the probability of the outcome $O$ given that $A$ is executed, times the probability that $A$ is actually executed (i.e. the execution makes it to $A$ without failure or without taking another branch). Formally:

$$P(O) = P(O | A) P(A)$$

If this probability exceeds a given probability threshold $T$ then an attempt is made to repair the outcome.

In addition, it may be useful to place a horizon limit on the search under the assumption that one does not need to prepare too far in advance for many uncertain outcomes. A more sophisticated alternative is to use some sort of discount factor to discount actions further in the future. This approach tends to focus the repair effort on the most time critical and highest probability outcomes on the assumption that later outcomes are less likely and less pressing.

**Discussion**

**Status**

Although we have developed a preliminary implementation of the techniques described in this paper, our implementation is far from complete. We initially chose to use the LPG-TD planner of Gerevini, Saetti, & Serina (2006) for our deterministic planner because it can optimize plans based on a metric, which is important for generating high probability seed plans and precautionary repairs. Our implementation converts PPDDL domains to PDDL domains as described in Algorithm 2. It then uses LPG-TD to generate a seed plan. A progressive greedy scan of the seed plan is done to examine the alternative outcomes for actions in the seed plan. An outcome is examined if the probability that the outcome will occur exceeds a given threshold. In this case, LPG-TD is called again to determine if an alternate plan to the goal exists for this outcome. If a plan is not found, then an unrecoverable outcome has been found and Algorithm 4 is used to try to find a precautionary or conformant solution that avoids the dead end. When LPG-TD returns a plan, this plan is used as a basis for the repair. If the probability of reaching the goal with this new plan is less than the threshold, that plan is recursively scanned. A horizon is used to limit the number of steps scanned in the plan and the number of levels of recursion during the repair algorithm. Once the plan has been analyzed up to the horizon, execution begins. After each step has executed, the analysis step is repeated, taking into account the new state of the world.

There are a number of deficiencies in our implementation. First, it does not yet include the ability to do confrontation as described in Algorithm 3, and we recognize that this is critical in some situations. Second, we call LPG-DT in order to determine whether or not each action outcome is unrecoverable. As we noted earlier, this becomes expensive as the number of actions in the plan with uncertain outcomes increases. We believe that recognition of reversible outcomes and use of plan graphs to recognize unrecoverable outcomes is essential to making this process efficient and practical. Despite these deficiencies, we have attempted some initial tests on simple problems from the first two probabilistic planning competitions. In most of these domains, actions are reversible, so run-time replanning alone is sufficient, and precautionary planning provides no additional benefit. In the few domains where unrecoverable outcomes are possible, precautionary planning can help – we have been able to achieve the goal in a larger percentage of problems than is possible with only run-time replanning. However, because of our reliance on LPG-DT to examine each outcome, our current implementation is slow. We are currently working on remedying these deficiencies.

**Related Work**

As we mentioned in the introduction, this work is inspired by the somewhat surprising success of FF-rePlan in the probabilistic tracks of the 2004 and 2006 International Planning Competitions (Younes et al. 2005; Bonet & Givan 2006). FF-rePlan plans optimistically, considering only the most likely effect of each action, then monitors execution and replans when failure occurs. Because FF-rePlan considers only the most likely outcome for each action, there may be many lower probability plans that it cannot find. Furthermore, FF-rePlan makes no attempt to optimize – that is, it does not consider action probabilities in its search for plans. It may therefore settle for a low probability plan when a higher probability plan is readily available.

Concurrent with our work, Jiménez, Coles, & Smith (2006) also recognized that it was possible to trick a deterministic planner like LPG into searching for unconditional plans of high probability by 1) including a separate deterministic action for each possible probabilistic action outcome, and 2) assigning a cost to each such deterministic action equal to the negative loga-
that much of the uncertainty in practical domains in-
et al.
the original work on Incremental Contingency Planning
A primary motivation for Continuous Uncertainty
future work
comes.
achieve goals, this is secondary to avoiding bad out-
bles somewhere in the future. While it will attempt to
the next action that it takes will not get it into trou-
the failure. The process is anytime in nature and incremen-
tally improves the schedule as long as time and com-
putational resources permit. ICP takes a similar ap-
proach to planning under uncertainty. A seed plan is
first constructed using a deterministic planner. Monte
Carlo simulation is then used to identify possible fail-
ure points. Heuristics are used to de-
cide which failure point to consider, and a deterministic
planner is used to construct a conditional branch for the
failure. Again, the process repeats as long as time and
computational resources permit. It is worth noting that
both of these approaches were developed to deal with
more difficult problems involving oversubscription and
goal utilities, and uncertainty in action duration and
resource usage. As a result, the heuristics and meth-
ods for choosing branch points and branch conditions
are considerably more complicated than what we have
considered here. One difference between Precau-
tionary Planning and the JIC and ICP work is that nei-
ther of those techniques were capable of inserting pre-
cautious steps – i.e. modifying that portion of the
plan or schedule prior to the branch point. This turned
out to be a significant weakness in the application of
ICP to practical problems. However, the most impor-
tant distinction between Precautionary Planning and
previous incremental approaches is our combination of
incremental improvement with replanning. This com-
bination fundamentally changes the focus of the plan
improvement effort so as to concentrate on only those
outcomes that lead to dead ends. This makes a big dif-
fERENCE in the heuristics and search strategy, and in
the robustness of the resulting system to uncertainty.

Finally, the CIRCA system (Musliner, Durfee, & Shin 1993) also has some similarities with Precau-
tionary Planning. Basically, CIRCA is a real time con-
trol system that attempts to look ahead and avoid any pos-
sibility of bad outcomes. In effect, it tries to prove that
the next action that it takes will not get it into trou-
ble somewhere in the future. While it will attempt to
achieve goals, this is secondary to avoiding bad out-
comes.

Future Work
Continuous Uncertainty A primary motivation for
the original work on Incremental Contingency Planning
(Bresina et al. 2002; Dearden et al. 2003) was the fact
that much of the uncertainty in practical domains in-
volves uncertainty in the duration and resource usage
of actions – in other words, uncertainty about contin-
uous quantities. For problems like this, uncertain
actions do not have a finite set of discrete outcomes, so
traditional MDP approaches are not adequate without
first discretizing the uncertain outcomes. Although this
paper has been focused on traditional planning under
uncertainty where actions have discrete uncertain out-
comes, the general technique is aimed at a broader class
of problems involving uncertainty in continuous quan-
tities. As in the previous work on incremental contin-
ergy planning, an initial seed plan would be generated
using the expected behavior of the actions. Given a seed
plan and an action with an uncertain continuous out-
come, the challenging problems are 1) to figure out the
range of outcomes that are likely to cause the current
seed plan to fail, and 2) determine whether the plan can
be repaired for this range of outcomes. In essence, we
want to find the subset of the range of outcomes that
are unrecoverable (or have low probability of recovery)
and apply the techniques of this paper to further re-
duce the range of those outcomes. In other words, for
continuous uncertainty the real objective is to reduce
the range and probability of unrecoverable outcomes,
rather than to completely eliminate them.

Oversubscription and Goal Values Another issue
that our framework does not currently address is that
of oversubscription and goal values. In our example we
had the simple goal of being across the border – there
was no value associated with this goal, and no costs
associated with actions. If the value of the goal were
low, and the cost of getting the spare tire were high, the
optimal plan might be to abandon the goal if a flat tire
occurs. Unfortunately, when actions have costs, and
goals have values, our simple approach of tricking a de-
terministic planner into finding good (high probability)
plans no longer works. The problem is that each action
outcome now has a cost, a probability of success, and
some expected benefit. The trouble is that 1) it is not
clear how to assign a meaningful benefit (distribute util-
ity) to individual actions, and 2) it is not clear how to
combine these quantities into a single “cost” that can
be used for optimization. In order to deal with costs
and utilities one could start with a more sophisticated
planner that is capable of dealing with goal utilities and
oversubscription problems. However, in this case it is
not clear how to take probability into account. One
could potentially penalize low probability outcomes by
adding a cost to each deterministic instance of the ac-
tion based on the negative logarithm of the outcome
probability. Using such an approach, a deterministic
oversubscription planner could be used to generate seed
plans.

Once a seed plan is generated, there is a second prob-
lem of deciding which outcomes to repair. In this case,
one is not just concerned with whether or not the goal
can be reached from an action outcome, but with how
much utility will be lost if the outcome occurs. Gener-
ally, more complex techniques such as those described in Dearden et al. (2003) may be required in this case.

Conclusion

One can argue that the primary reason FF-rePlan did so well in the probabilistic planning competitions is that the domains and problems had very few dead-end outcomes – that is, outcomes where it was no longer possible to reach the goal(s). Although there is much truth to this observation, we believe there is still an important lesson to be learned; *many action outcomes can be dealt with efficiently by run-time replanning*. It is only those outcomes that would lead to failure that a planner really needs to worry about ahead of time. Thus, we think it is a mistake to consider either contingency planning or replanning in isolation. Full contingency planning (policy generation) is too difficult and too slow, and replanning alone is not robust enough. Any system capable of dealing with significant problems will need a combination of the two approaches – replanning to deal with all the annoying but relatively harmless outcomes, and contingency planning to deal with those outcomes that would result in failure.

We have presented a framework for dealing with planning under uncertainty that interleaves planning and execution. We assume a model where actions may have probabilistic outcomes, the execution agent can observe the state of the world after each probabilistic action completes, and there is time for replanning after such observations are made. Under our framework, contingency planning is done to prevent unrecoverable failures, and replanning is done when recoverable failure occurs. This is different from both contingency planning and replanning. In contingency planning, attempts are made to prevent all failure, whereas we only concern ourselves with unrecoverable failure. In replanning, a dead end occurs when an attempt is made to replan after an unrecoverable failure. Our planning framework does contingency planning to avoid this. Another key to our framework is that we use a fast deterministic planner for plan generation. We have developed algorithms for using a deterministic planner to 1) generate plans that have a high probability of success and 2) to insert precautionary actions and generate contingency branches for unrecoverable action outcomes. We have also described several heuristics for discovering unrecoverable outcomes.

Acknowledgements

We would like to thank Alfonso Gerevini, Alessandro Saetti, Ivan Serina, Paolo Toninelli for making an executable of LPG-td available and for helpful correspondence. We would also like to thank Minh Binh Do and J. Benton for making the code of SAPA available and for quickly answering questions related to this code. This work was supported by the NASA Harriett Jenkins Predoctoral Fellowship Program, and by the Automation for Operations project of the NASA ETDP program.

References

Bonet, B., and Geffner, H. 2003. Labeled RTDP: Improving the convergence of real-time dynamic programming, In 13th Int. Conf. on Automated Planning and Scheduling (ICAPS-03), 12–21.


