Extended Abstract: Risk-Sensitive Planning with Dynamic Uncertainty

Liana Marinescu
Department of Informatics, King's College London
liana.marinescu@kcl.ac.uk

Publications produced

To cite the heuristic described in this paper, please refer to:

Heuristic Guidance for Forward-Chaining Planning with Numeric Uncertainty (Marinescu and Coles 2016).

1 Introduction

Many compelling applications of planning arise from scenarios that are inherently uncertain. In some cases it is possible to adequately capture the dynamics of the world without modeling uncertainty, and thus to employ classical planning techniques. However, in many other cases it is impossible to ignore uncertainty without hindering the planner's knowledge about the world, and hence obtaining sub-par solutions.

Our research on uncertainty so far is twofold:

- Improving *heuristic guidance* for problems with numeric uncertainty, by including relevant probabilistic information in the heuristic (with a negligible computational cost).
- Extending prior work on *building a policy offline* for problems with nondeterministic action outcomes (Muise, McIlraith, and Beck 2012), by allowing it to support continuous numeric uncertainty.

The first contribution is focused on finding plans for models where there is uncertainty in the outcomes of numeric effects (each governed by a continuous probability distribution). The task is to find a plan where all the preconditions are met, and the goals are reached, with some confidence θ. This paradigm has been explored by previous work, e.g. (Beaudry, Kabanza, and Michaud 2010; Coles 2012); our first contribution is in providing effective *heuristic guidance* in such a setting.

The second contribution – extending prior work on propositional uncertainty to numeric uncertainty – revolves around offline planning. The task is to *build a policy offline* for models where action outcomes have a set of discrete, nondeterministic effects. The class of propositional problems has been addressed by Muise (Muise, McIlraith, and Beck 2012), but numeric uncertainty still remain a challenge. We have extended one of the basic mechanics of policy-building – that of regression (applying an action "backwards" in a state, to obtain a sufficient predecessor state) – from the propositional case, to the case of independently distributed Gaussian numeric uncertainty.

2 Background: Continuous Uncertainty on Actions' Outcomes

To begin with, we build on the state-progression semantics of the planner RTU (Beaudry, Kabanza, and Michaud 2010). Actions have propositional and numeric preconditions and effects, as in classical numeric planning, but the numeric effects have outcomes that are drawn from probability distributions. We say that each effect is of the form $\langle v \ op \ D(\mathbf{v})\rangle$ where $op \in \{\text{+=}, =\}$ and D is a (possibly deterministic) probability distribution that governs the range of outcomes of the effect. For example:

- $\langle battery \ \text{+=} \ \mathcal{N}(-10, 2^2)\rangle$ – decrease battery by an amount with mean 10 and standard deviation 2.
- $\langle coal \ \text{+=} \ \mathcal{N}(15, 3^2)\rangle$ – increase coal by an amount with mean 15 and standard deviation 3.
- $\langle position\_error = \mathcal{N}(0, 1^2)\rangle$ – reset the position error to 0 with standard deviation 1 (e.g. after calibration).

In addition to this, we have a confidence level $\theta \in [0.5, 1)$: because numeric effects have uncertain outcomes, we need to prescribe how certain we must be that each numeric condition is satisfied.

A Bayesian network (BN) is used to define the belief of each $\mathbf{v}$, and as actions are applied, the network is updated with additional variables. In a state S_i, for each $v^j \in \mathbf{v}$, a variable v_i^j is associated with the belief of v. If an action a is applied, leading to a state S_{i+1}, then for each numeric effect $\langle v^j \ \ op \ \ D(\mathbf{v})\rangle$, two random variables are added to the network. The first of these, D_{i+1}^j, represents $D(\mathbf{v})$. The second, v_{i+1}^j, is associated with the belief of v in S_{i+1}, and it is determined by either:

- $v_{i+1}^j = v_i^j + D_{i+1}^j$, if *op* is +=;
- $v_{i+1}^j = D_{i+1}^j$, if *op* is =.

The BN is key to determining whether a plan meets the required confidence level θ. An action a is applicable in a state S_i if $Pre(a)$ is satisfied. A sequential (linear) solution is a sequence of steps $[a_0, .., a_n]$, implying a state trajectory $[I, S_0, .., S_n]$. We use the BN to ensure that with $P \geq \theta$, in a given execution of the plan, each action's preconditions are met and S_n satisfies any hard goals.

The state progression formalism of Beaudry *et al* was adopted and extended by Coles (2012) as the basis of an

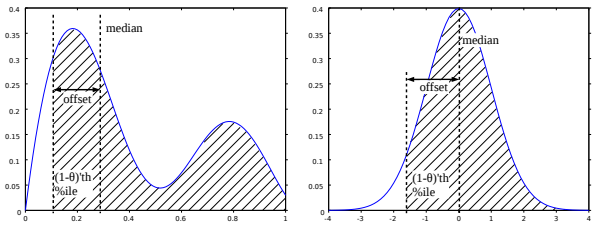


Figure 1: Possible probability distributions: Arbitrary (left) and Gaussian (right).

over-subscription planning approach. A forward-chaining planner following these semantics was used to find a single plan, onto which branches were added by making additional calls to the planner. A range of other approaches have been adopted for planning under uncertainty, such as those based on the use of Markov Decision Processes, e.g. (Meuleau et al. 2009; Mausam and Weld 2008; Rachelson et al. 2008); these approaches are particularly useful when a *policy* needs to be found. As our first contribution is on the heuristic inside a forward-chaining planner, our focus will be on planning under the semantics of RTU described above.

3 Heuristic Guidance for Forward-Chaining Planning with Numeric Uncertainty

3.1 Relaxing Numeric Uncertainty

In deterministic forward-chaining numeric planning, one way to guide search is the Metric Relaxed Planning Graph (RPG) heuristic (Hoffmann 2003). This performs a forward reachability analysis that estimates the number of actions needed to reach goals by relaxing the effects of actions. For numeric state variables, this amounts to estimating reachable bounds on the values of variables, by optimistically assuming that increase effects only increase the upper bound, and decrease effects only decrease the lower bound.

When working with RTU's semantics, Coles (2012) adapted this to assume for heuristic purposes that each variable takes its *median* value. From Jensen's inequality, we know that if $\theta \geq 0.5$, this is guaranteed to be a relaxation. However, as θ becomes large, it also means the heuristic is increasingly unrealistic: a numeric condition might be true assuming variables take their median values; but not when accounting for the uncertainty in their values. In this section, we will present two strategies that improve on this:

- we incorporate the shape of the distribution on variables' values in the heuristic evaluation, rather than discarding it and using the median;
- for Gaussian distributions, we explicitly track the uncertainty of variables in the relaxed planning graph.

Heuristic Guidance with Monotonically Worsening Uncertainty Uncertainty can affect problems in two ways: it either gets worse monotonically (error accumulates and no action can rectify it); or it may be purposefully corrected (there may be actions that reduce the error, such as recharging batteries to a fixed value, or visiting a precise weighing station).

We first discuss the case of monotonically worsening uncertainty. Outside the heuristic, each precondition is of the form $\mathbf{w}.\mathbf{v} \geq c$, and a Monte Carlo simulation is used to estimate the probability distribution of $\mathbf{w}.\mathbf{v}$. Using this distribution, we can test whether the condition is satisfied with probability θ, i.e. whether the $(1-\theta)$'th percentile of $\mathbf{w}.\mathbf{v}$ is $\geq c$. We represent this percentile as follows:

$$p_{1-\theta}(\mathbf{w}.\mathbf{v}) = median(\mathbf{w}.\mathbf{v}) - \mathit{offset}_\theta(\mathbf{w}.\mathbf{v})$$

In effect, offset_θ is the margin of error that must be tolerated, for the precondition to be true with probability θ. We illustrate the intuition behind this margin in Figure 1. The condition itself can then be rewritten:

$$median(\mathbf{w}.\mathbf{v}) \geq c + \mathit{offset}_\theta(\mathbf{w}.\mathbf{v}) \quad (1)$$

We define that uncertainty is *monotonically increasing* if offset_θ *can never decrease*. In this case, it is still a relaxation to use the offset values when determining which preconditions are true in the heuristic – the only way to make the condition true would be to apply actions that affect the values of $\mathbf{v}$, as no actions that decrease offset_θ exist.

An illustrative example would be an autonomous car with a certain amount of fuel, which is used gradually until it runs out; refueling is not possible. The activities performed by the car (e.g. start engine, accelerate, stand still, park) each require fuel, but the amount varies non-deterministically. As the plan is constructed, uncertainty and hence offset_θ accumulates monotonically. We can thus heuristically evaluate a state by assuming offset_θ is constant, and takes its current value; this is guaranteed to be a relaxation, as offset_θ can never become smaller.

Heuristic Guidance with Gaussian Uncertainty So far, we explained how to incorporate distributions on the left-hand side of preconditions ($\mathbf{w}.\mathbf{v}$) into heuristic computation, by using the offset_θ value to capture uncertainty information. The relaxation holds when error accumulates and cannot be lowered. However, problems may contain actions such as *recharge-batteries* or *visit-weigh-station*, which reduce uncertainty.

The challenge in these sorts of problems is to ensure the heuristic remains a relaxation. This is possible in a useful subset of domains, where the uncertainty is due to independent Gaussian-distributed effects on variables, and therefore has an analytic form. We can utilize this form and extend the Metric RPG to additionally track the variance on each variable, $\sigma^2(v)$. The expansion phase, building the RPG, proceeds as follows:

- For each variable $v \in \mathbf{v}$, we track the upper and lower bound on its median value. In the first RPG layer, these are equal to the value of v in the current state S. We additionally track $\sigma^2(v)$, the variance on v. In the first RPG layer, this is the value according to the BN for S.
- In a regular RPG, if a numeric effect is applied that increases (decreases) some $v \in \mathbf{v}$, the upper (resp. lower) bound on v at the next fact layer is updated accordingly. Now, additionally, if a numeric effect decreases $\sigma^2(v)$, the lower bound on $\sigma^2(v)$ at the next fact layer is decreased[1].

[1]Effects increasing $\sigma^2(v)$ are ignored. If $\theta \geq 0.5$, adding more

Algorithm 1: RPG Solution Extraction

Data: RPG, a relaxed planning graph
Result: p, a relaxed plan
1 $last \leftarrow$ last layer index in RPG;
2 $goals[last] \leftarrow G$ *(i.e. the problem goals)*;
3 **for** $l \in [last..0]$ **do for** $(\mathbf{w.v} \geq c) \in goals[l]$ **do**
4 $\quad prev \leftarrow$ max value of $\mathbf{w.v}$ in layer l-1;
5 $\quad prev\_\sigma^2 \leftarrow$ min value of $\sigma^2(\mathbf{w.v})$ in layer l-1;
6 $\quad prev\_offset_\theta \leftarrow prev\_\sigma.\Phi^{-1}(\theta)$;
7 $\quad$ **if** $prev \geq c + prev\_offset_\theta$ **then**
8 $\quad\quad$ add $(\mathbf{w.v} \geq c)$ to $goals[l\text{-}1]$; **continue**;
9 $\quad$ **for** $(w.v) \in \mathbf{w.v}$ *where* $w \neq 0$ **do**
10 $\quad\quad$ Choose actions from l-1 that increase $(w.v)$;
11 $\quad\quad$ Add them to the relaxed plan and subtract their effects from c;
12 $\quad\quad$ **if** $prev \geq c + prev\_offset_\theta$ **then break**;
13 $\quad$ **if** $prev \geq c + prev\_offset_\theta$ **then**
14 $\quad\quad$ add $(\mathbf{w.v} \geq c)$ to $goals[l\text{-}1]$; **continue**;
15 $\quad max\_offset \leftarrow prev - c$;
16 $\quad max\_\sigma^2 \leftarrow (max\_offset/\Phi^{-1}(\theta))^2$;
17 $\quad$ add $(-\sigma^2(\mathbf{w.v}) \geq -max\_\sigma^2)$ to $goals[l]$;
18 $\quad$ add $(\mathbf{w.v} \geq prev)$ to $goals[l\text{-}1]$;

- To decide which actions are applicable in each layer, we take variance into account when checking precondition satisfaction, as follows. For a precondition of the general form $\mathbf{w.v} \geq c$, we can use the additive properties of Gaussians to compute the variance of $\mathbf{w.v}$:

$$\sigma^2(\mathbf{w.v}) = \sum_{w.v \in \mathbf{w.v}} w^2.\sigma^2(v)$$

We obtain the offset using the Gaussian quantile function:

$$offset_\theta(\mathbf{w.v}) = \sigma(\mathbf{w.v}).\Phi^{-1}(\theta)$$

Hence, from Equation 1, the precondition becomes:

$$median(\mathbf{w.v}) \geq c + \sigma(\mathbf{w.v}).\Phi^{-1}(\theta)$$

This gives us everything we need to build an RPG. We can be confident that the $offset_\theta$ values used are relaxations, because smaller values of variance result in smaller values of the Gaussian quantile function Φ^{-1}; and the semantics of the RPG guarantee we will underestimate variance.

The next step is to extract a relaxed plan from the RPG; we illustrate this in Algorithm 1. The first thing to note is on lines 5 and 6, where we compute the $offset_\theta$ necessary for the condition to be met. Actions are then chosen in the standard way to attempt to meet the precondition, given this value of $offset_\theta$. Then, if line 13 is reached and the precondition is still not true, it must mean that a decrease in variance caused it to become true at layer l (having been false at layer l-1). We now need to choose actions that decrease variance enough to achieve this. On line 15, we work out what $offset_\theta$ needs to be reduced to in order to make the precondition true; we then compute its corresponding variance on line 16. This variance can then be used to construct a new condition to be satisfied at this layer: this causes actions to be added to the relaxed plan in order to reduce variance on a later iteration of the loop.

As a result of the algorithm described above, the relaxed plan now contains uncertainty-reducing actions. This makes for a better-informed heuristic, which is able to provide improved guidance and dead-end detection to the search, as will be demonstrated in the following section.

uncertainty never contributes towards preconditions becoming true, so it suffices to track only the smallest reachable values of variance.

3.2 Evaluating the new Heuristic

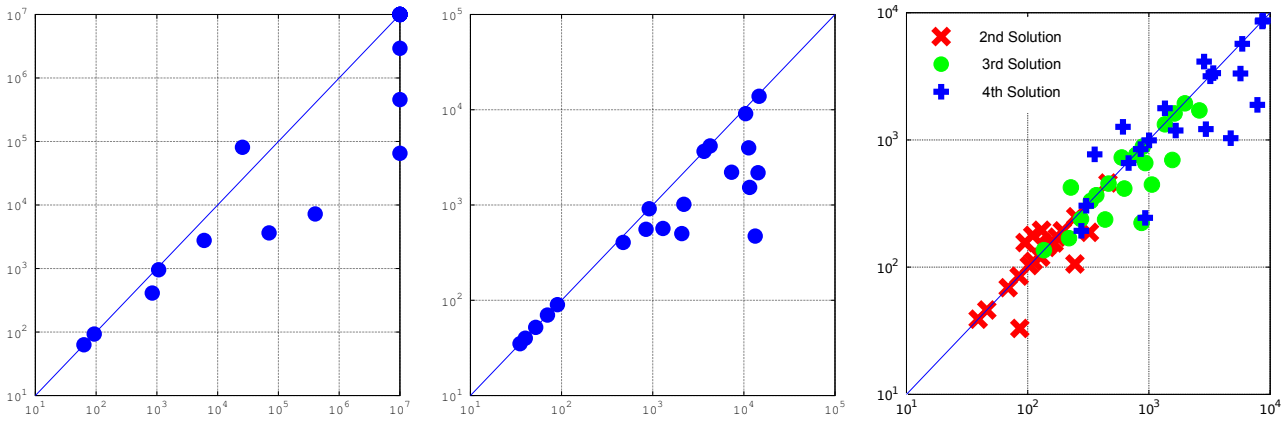
We evaluate on three domains: Rovers and AUV from (Coles 2012); and a variant of TPP from (Gerevini et al. 2009). In Rovers, the activities of a planetary rover are constrained by battery usage, which has Gaussian uncertainty, and the battery can only be recharged at certain locations. In TPP, the domain is modified to model the acquisition of sufficient amounts of bulk materials (e.g. coal), and trucks can visit weighing stations at some suppliers to top up or shed excess load, which reduces uncertainty. AUV is an over-subscription problem where the activities of an underwater vehicle must be planned with a strict bound on total time taken, and with normally distributed activity durations. Tests were performed on 3.5GHz Core i5 machines with a limit of 4GB of memory and 1800s of CPU time.

Overall, the new heuristic leads to faster planner performance; the time-to-solve scatterplots look the same as the nodes-generated scatterplots in Figure 2. The extra computational work (tracking variances etc.) does not adversely affect the time taken to heuristically evaluate a state. Thus, because significantly fewer states need to be evaluated, and state evaluation times are comparable, the performance of the planner is significantly better.

For the Rovers domain (Figure 2a), most striking are the points on the far right of the graph – these indicate problems that were previously unsolvable but can now be solved. In part, this is because the new heuristic is able to recognize many more states as being dead ends, because it does not disregard uncertainty on the battery level when evaluating preconditions. In contrast, by ignoring uncertainty, the old relaxed plans relied on moving somewhere to recharge, even though in reality uncertainty made it impossible for that move action to be applied. The new heuristic often avoids this pitfall by accounting for uncertainty to a greater extent.

In TPP (Figure 2b), all the problems could be solved by both the old and the new heuristic. However, by not accounting for uncertainty, the old heuristic can reach states in which the relaxed plan does not need to buy any more of any goods. In these states, the heuristic value is 0. As acquiring additional goods requires combinations of travel and buy actions, a substantial amount of search must be performed with no effective heuristic guidance. Unlike Rovers, there are no dead ends due to these travel actions, so this blind search will succeed, but is very time consuming – in problems furthest from the line $y = x$, the majority of nodes evaluated have an old heuristic value of 0.

AUV is an over-subscription problem: search reports a solution plan every time it finds one that solves more goals than the best so far. We are hence interested in the search



(a) Rovers (b) TPP (c) AUV

Figure 2: Nodes generated to solve problems in the three evaluation domains. Axes are logarithmic, comparing prior work (X axis) with the new heuristic (Y axis). The Two-Tailed Wilcoxon Signed-Rank Test confirms results are significant to $P \geq 0.95$.

effort to find progressively better solutions. Figure 2c compares the nodes generated by each configuration to find the 2nd, 3rd and 4th solutions. (These correspond to satisfying 1, 2, and 3 goals respectively.) The relaxed plans produced by the old heuristic, by ignoring uncertainty, more often use actions that there is actually no time to complete. Disregarding uncertainty is less of an impediment than in Rovers and TPP, as there is no scope for planning actions that reduce uncertainty (unlike battery charge or goods purchased, actions cannot create more time). Nonetheless, the new heuristic is generally able to find better solutions more quickly. If left to run for long enough, search with the old heuristic will tend to find solutions as good as search with the new heuristic, but loses out earlier in the search.

As a concluding remark for our results, we note that so far we assumed $\theta = 0.99$. At $\theta = 0.8$, the improvements from using the new heuristic are still noticeable, but not as substantial. By $\theta = 0.6$, which is close to the median ($\theta = 0.5$), there is no statistically significant difference between the two, as uncertainty has only a modest effect on the heuristic, or indeed search itself. This confirms that our heuristic meets our headline aim of being able to better guide the planner when the consequences of uncertainty bear a significant effect upon what is a reasonable solution plan.

4 Background: Discrete Uncertainty on Actions' Outcomes

The next step in our research on uncertainty revolves around problems with discrete non-deterministic action outcomes. In order to reason about discrete non-determinism, we use actions that have a precondition (as in the deterministic case), but instead of having a single effects list, they have several. Applying an action will nondeterministically trigger one of these. We assume states are *fully observable* – we can observe what effects an action had. A solution to problems containing such actions can be represented by using a policy – a set of rules that dictates what should be done in each state.

Since action outcomes are discretely different, a policy may need to branch out and apply different actions in the different states reached. A weak plan corresponds to a single trajectory of actions that leads from the initial state to a goal state, assuming it is possible to choose which action outcome occurs at each point. Weak plans can be found using a deterministic planner given as input the *all outcomes determinisation* (Yoon, Fern, and Givan 2007) – the representation of an action with discrete outcomes as a set of actions having identical preconditions but different lists of effects (one list corresponding to each discrete outcome).

Muise *et al.* (2012) present an approach where, by repeatedly invoking a deterministic planner to find weak plans, it is possible to incrementally build a policy. Key to the success of their approach is exploiting *relevance* – by regressing the goal through a weak plan step-by-step, they determine which facts at each point are relevant to plan success.

Regression begins from the goals, which here are a set of propositions, ps'. Regressing ps' through an action A, with preconditions $pre(A)$ and add effects $add(A)$ yields a partial state ps where:

$$ps = (ps' \setminus add(A)) \cup pre(A)$$

A policy can then be built from these pairs, each $\langle ps, A \rangle$. If executing the policy reaches a state where $S \vDash ps$, then A is applied. Otherwise, if there is no such match, the planner is invoked from S, producing another weak plan which is added to the policy in this way. Policy building terminates when $\exists \langle ps, A \rangle . S \vDash ps$ for all states S reachable from the initial state, via the action choices indicated by the policy.

One caveat of this process is that it must be restarted if a dead-end state D is reached. If this happens, Muise *et al.* record *forbidden state–action pairs*: D is regressed through all actions $[a_0..a_n]$ that could lead to it, yielding states $[S_0..S_n]$; then each $\langle S_0, a_0 \rangle$ is forbidden, as applying a_i in S_i would reach D once again. To improve generality, each state S_i is *generalised* – a greedy algorithm is used to remove facts that do not affect whether S_i is a dead-end.

The policy produced by this process, overall, is a strong cyclic plan: it will always reach the goals, if the goals can be reached. The work has since been developed to support conditional effects (Muise, McIlraith, and Belle 2014) and sensing actions (Muise, Belle, and McIlraith 2014), which

are interesting avenues we hope to look at in the future. For now, we concern ourselves with extending this well-defined propositional formalism to incorporate continuous numeric uncertainty.

5 Building Policies Offline for Continuous and Discrete Uncertainty

5.1 Allowing Numeric Uncertainty in Action Effects

As detailed in Section 3, we have a planner kernel that is capable of supporting actions with continuous numeric uncertainty. Additionally, as detailed in Section 4, Muise *et al.*'s work on offline policy-building currently supports propositional effects. It is therefore a natural step to use our planner to extend their work to numeric effects, thus allowing a richer set of problems to be tackled.

Furthermore, as we will describe below, extending previous work in this manner provides more than just the ability to build policies with numeric effects. It retains and generalises the mechanics we defined previously for dealing with numeric uncertainty. This greatly increases the level of realism that problems can reflect, as continuous uncertainty (e.g. will we step 10 metres or 10.5 metres?) can now function alongside discrete uncertainty (e.g. will we step forward or will we blow a fuse?).

5.2 Representing Partial States with Numeric Constraints

The approach of Muise *et al.* targets problems with propositional constraints. In their case, partial states can be intuitively defined as containing only a subset of propositional constraints (e.g. the full state {at-home, have-package, is-birthday} satisfies the partial state {at-home}). We extend this intuition to numeric constraints: if a full state contains the constraints $\{battery{=}15, altitude{=}40\}$, then this can satisfy a partial state $\{battery \geq 10\}$. But, we must additionally take care to account for any variance on the value of $battery$ at this point.

We first define a representation of a constraint that includes an explicit record of any variance that needs to apply to it. For a constraint $\mathbf{w}.\mathbf{v} \geq c$ we record a constraint tuple $\langle ft, c, vt, av \rangle$ where:

- ft are the formula terms, initially the weighted-sum of variables, $\mathbf{w}.\mathbf{v}$.
- c is the right-hand-side constant, initially c.
- vt are variance terms. These are initially $\{\langle w_0, \sigma^2(v_0), 0\rangle..\langle w_n, \sigma^2(v_n), 0\rangle\}$ for each $w_i.v_i \in \mathbf{w}.\mathbf{v}$.
- av is an accumulated variance value, initially 0.

Performing regression is akin to applying an action "backwards" in a state, to obtain the sufficient conditions for that action application to have resulted in that state. If a constraint C is regressed through a numeric effect $v{+}{=}\mathbf{w}'.\mathbf{v}'+e$, where $w.v \in ft$, the resulting constraint C' is:

$$\begin{aligned} ft' = &\quad ft + w.(\mathbf{w}'.\mathbf{v}') \\ c' = &\quad c - w.e \\ vt' = &\quad vt \\ av' = &\quad av \end{aligned}$$

Regressing through the effect $v = \mathbf{w}'.\mathbf{v}' + e$ gives:

$$\begin{aligned} ft' = &\quad ft - w.v + w.(\mathbf{w}'.\mathbf{v}') \\ c' = &\quad c - w.e \\ vt' = &\quad vt \\ av' = &\quad av \end{aligned}$$

Effects on variance also affect the variance portions of constraints. If $\sigma^2(v)$ is changed by an effect $\sigma^2(v){+}{=}e$ then for any constraint with an element $\langle w, \sigma^2(v), k\rangle \in vt$, vt' is identical modulo k being increased by e. For the effect $\sigma^2(v) = e$, then for any constraint with an element $\langle w, \sigma^2(v), k\rangle \in vt$, vt' is identical modulo this element being removed, and $av' = av + e.w^2$. Conceptually, after regressing through this effect assigning variance a fixed value, any earlier effects on variance are moot: it suffices to transfer the newly assigned (and weighted) amount to the accumulated variance.

To determine if a state S satisfies a constraint C with confidence θ, we define the Gaussian distribution with mean ft (taking values of variables from S), and with variance:

$$av + \sum_{\langle w, \sigma^2(v), k\rangle \in vt} w^2.(S[\sigma^2(v)] + k)$$

Then, if the $1-\theta$'th percentile of this Gaussian is $\geq c$, we say S satisfies C. Effectively, the condition must be true allowing for the variance in S, and any in the condition itself. This is a slight departure from checking preconditions on actions as described in Sections 2 and 3, where the only variance to account for was that in S. Here, the constraints denote preconditions that must be true *later in the weak plan*, so the variance within the constraint tracks the impact of effects on variance between S and when the precondition must in fact hold.

5.3 Handling Dead Ends

Muise *et al.* introduced a particularly insightful contribution concerning dead ends. As mentioned in Section 4, their idea is to mark partial state–action pairs as forbidden if they lead to a dead end. Whenever a new dead end is found while building the policy, its corresponding partial state–action pair (i.e. the pair that would lead to that dead end) is generated via one step of regression, and stored in the forbidden list. The policy is then deleted, and policy building restarted with this new information at hand.

We build our extension along the same lines. As we are performing numeric planning, our states are split into two parts: logical and numeric. The former can be handled exactly as by Muise *et al.*: regressed through the logical effects of the action. For the latter, for each variable value $v{=}k$ in the state, following the representation set out in Section 5.2 we build pairs of tuples $\langle v, k, \{\}, 0\rangle$, $\langle -v, -k, \{\}, 0\rangle$. (Note this equally applies to variance-tracking variables – which are first-class citizens and appear as state variables along with the others.) These numeric constraints are then regressed through the numeric effects of the action.

To generalise these dead-ends, we again look at the logical and numeric parts of the state. For the logical, the approach of Muise *et al.* based on the RPG heuristic can be used: in Section 3, we fortuitously described exactly the sort

of RPG heuristic that would be needed to support this. For the numeric case, if static analysis reveals that larger/smaller values of a variable are better in terms of satisfying conditions, then we need only keep one constraint tuple, not a pair. Simply, for a dead-end state where v=k, if we know larger values of v are better, we only need record $\langle -v, -k, \{\}, 0\rangle$ (which is analogous to $v \leq k$), as any state with this value of v or worse is going to be a dead end.

5.4 Preliminary Observations

So far, we have conducted some preliminary tests on a modified version of the rovers domain, extended to better model the distribution of energy usage when moving. Rather than assuming this can be adequately captured by a single Gaussian-distributed variable, there is a pessimistic outcome with high energy usage corresponding to a failure mode of the rover; and an optimistic outcome corresponding to an unusually clear path. This leads to a policy being built that considers what to do for each of these outcomes.

It is clear at this point that the power of partial states seen in the propositional case, is also being seen in the numeric case. If a pessimistic navigate outcome occurs, then the policy does not need to branch if there is still enough power left; i.e. the resulting state still satisfies the relevant partial state. Even if branching does occur (i.e. the planner needed to be invoked to find another weak plan) the recharge actions have an interesting effect. As these recharge the battery and clear the variance on its value, there are no restrictions on battery charge in the partial state before them. Thus, even if the policy splits into distinct branches, these often later merge.

Generalising dead-ends for numeric resources also has a beneficial effect: in plain English, the dead-ends seen can generally be interpreted as 'if the rover is in this location, with only this much battery charge, or too much variance on battery charge, then applying this action will lead to a dead-end'. These situations arise when the rover would be unable to reach somewhere it can recharge and then resume operations. When restarting policy building, the dead-ends found are effective in pruning unsuitable action choices from search.

6 Future work

So far we have successfully improved heuristic guidance in problems with numeric uncertainty. We are also well into providing an extension to propositional offline policy-building, allowing it to work in problems with numeric action effects. The next stage consists of finalising this extension, and conducting a detailed evaluation.

We will also allow our planner to accept domains where there is non-Gaussian uncertainty. For this, we would generalise the techniques we already implemented and tested, in order to make the transition from Gaussian probability distributions to arbitrary probability distributions. This would considerably broaden the spectrum of problems the planner will be able to solve. In particular, we intend to incorporate Bayesian Network techniques to support non-Gaussian uncertainty, and adapt regression and probabilistic dead ends for this type of uncertainty.

The next promising avenue to explore afterwards would be to apply our technique to a "planning in the loop" setting, with the scope for adjusting the probability distributions of action outcomes based on experience gathered from plan execution so far. As a practical case study, we plan to test our approach on an autonomous robotic platform operating in a dynamic environment. A concrete example would be a quadcopter conducting a rescue mission inside a building that is still collapsing, or seeking out an outdoor waypoint amongst vegetation or foliage. The success of practical testing would indicate the programme of work has had its desired outcomes: developing planning approaches that function well with more realistic world dynamics, and broadening the spectrum of possible applications of planning.

References

Beaudry, E.; Kabanza, F.; and Michaud, F. 2010. Planning with Concurrency under Resources and Time Uncertainty. In *Proceedings of ECAI*.

Coles, A. J. 2012. Opportunistic Branched Plans to Maximise Utility in the Presence of Resource Uncertainty. In *Proceedings of ECAI*.

Gerevini, A.; Long, D.; Haslum, P.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic Planning in the Fifth International Planning Competition: PDDL3 and Experimental Evaluation of the Planners. *Artificial Intelligence*.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating Ignoring Delete Lists to Numeric State Variables. *Journal of Artificial Intelligence Research* 20.

Marinescu, L., and Coles, A. 2016. Heuristic guidance for forward-chaining planning with numeric uncertainty. In *Proceedings of ICAPS*.

Mausam, and Weld, D. S. 2008. Planning with Durative Actions in Stochastic Domains. *Journal of Artificial Intelligence Research* 31.

Meuleau, N.; Benazera, E.; Brafman, R. I.; Hansen, E. A.; and Mausam. 2009. A Heuristic Search Approach to Planning with Continuous Resources in Stochastic Domains. *Journal of Artificial Intelligence Research* 34.

Muise, C.; Belle, V.; and McIlraith, S. 2014. Computing contingent plans via fully observable non-deterministic planning. In *Proceedings of AAAI*.

Muise, C.; McIlraith, S.; and Beck, J. 2012. Improved non-deterministic planning by exploiting state relevance. In *Proceedings of ICAPS*.

Muise, C.; McIlraith, S.; and Belle, V. 2014. Non-deterministic planning with conditional effects. In *Proceedings of ICAPS*.

Rachelson, E.; Quesnel, G.; Garcia, F.; and Fabiani, P. 2008. A Simulation-Based Approach for Solving Temporal Markov Problems. In *Proceedings of ECAI*.

Yoon, S.; Fern, A.; and Givan, R. 2007. FF-Replan: A baseline for probabilistic planning. In *Proceedings of ICAPS*.