Non-Deterministic Planning with Numeric Uncertainty

Liana Marinescu and Andrew Coles

Abstract. Uncertainty arises in many compelling real-world applications of planning. There is a large body of work on propositional uncertainty where actions have non-deterministic outcomes. However, handling numeric uncertainty has been given less consideration. In this paper, we present a novel offline policy-building approach for problems with numeric uncertainty. In particular, inspired by the planner PRP, we define a numeric constraint representation that captures only relevant numeric information, supporting a more compact policy representation. We also show how numeric dead ends can be generalised to avoid redundant search. Empirical results show we can substantially reduce the time taken to build a policy.

1 Introduction and Background

Planning accounting for uncertainty is relevant to many interesting real-world problems, for instance where the dynamics of the environment make it difficult to plan on the basis of actions having a single, predictable outcome. In this work we look at fully observable non-deterministic (FOND) planning, i.e., applying an action has several possible outcomes, but we can observe which occurs. The task of planning here is to find a policy (a mapping from states to actions) that dictates what to do in each state reachable from the initial state.

Additionally, within this setting, we allow numeric effects to have continuous uncertainty; specifically, we allow Gaussian-distributed (independent) effects on variables. Each effect is of the form \( \nu \cdot \Phi \cdot N(w \cdot v + k, \sigma^2) \), where \( \nu \) is some state variable; \( \Phi \in \{+,-,=\} \); \( w \cdot v \) is a weighted sum of state variables; \( k \in \mathbb{R} \); and \( \sigma^2 \in \mathbb{R} \) is the variance of the effect. In each state we store the mean and variance of each variable: \( v \) and \( \sigma^2(v) \). For a numeric precondition (or goal) of the form \( w \cdot v \geq c \), we first compute the variance affecting it (1). The precondition is then true with confidence \( \theta \) iff (2) is satisfied (where \( \Phi \) is the Gaussian cumulative distribution function).

\[
\begin{align*}
\sigma^2(w \cdot v) &= \sum_{w \cdot v} \cdot \sigma^2(v) \quad (1) \\
w \cdot v &\geq c + \sigma(w \cdot v) \cdot \Phi^{-1}(\theta) \quad (2)
\end{align*}
\]

Effectively, this computes an offset we need to add onto the precondition so that, even accounting for uncertainty, it remains true e.g. 99% of the time for \( \theta = 0.99 \). The task of planning is then to find a policy such that at each point, the preconditions of the action to be applied (or the goals) are true with a given confidence level \( \theta \).

Our work builds upon two recent strands of research: a heuristic for planning with continuous numeric uncertainty [5]; and a planner for propositional FOND domains, PRP [8, 7]. In PRP, policies are found by making repeated calls to a deterministic planning kernel, which finds weak plans that work assuming the action outcome can be chosen. To then obtain a policy that covers the other outcomes, this process is recursively applied for the other states that could be reached. Key to the success of PRP are two ideas:

First, the policy maps partial states (rather than fully-specified states) to actions. These partial states are found by regression: every time a weak plan is added to the policy, the goals are regressed through it step-by-step, and at each point a partial-state–action pair is added to the policy. The benefit of regression is that it collects the weakest preconditions needed for the tail of the plan to succeed – if a state in a weak plan contains literals that are irrelevant to the remaining steps, these literals will not be included in the partial state.

Second, for states that cannot reach the goal, dead end generalisation is used. In the propositional case, this amounts to a sensitivity analysis to check which literals in a state are causing it to be a dead end. If a dead end state \( S \) contains \( f \), and evaluating \( S \cup \{\neg f\} \) with the Relaxed Planning Graph (RPG) heuristic [4] indicates \( S \) is still a dead end, then the truth value of \( f \) is irrelevant and can be ignored. Any state found during policy-building that matches a known dead end can then be discarded, avoiding wasted search effort.

Adapting these ideas to the continuous numeric case requires a planning kernel (we use our recent work [5]) and suitable definitions of regression and dead end generalisation. These form the next two sections of this paper, followed by an empirical evaluation.

2 Regression for Numeric Constraints

As mentioned above, in PRP, the policy maps partial states to actions; these partial states are found by regressing through the weak plans given by the planning kernel. Regression begins with the goal state and steps through the actions in reverse order. From a partial state \( ps' \), regressing through an action \( a \) with propositional preconditions \( Pre(a) \) and add-effects \( Eff^+(a) \) yields a new partial state:

\[
ps = (ps' \setminus Eff^+(a)) \cup Pre(a) \quad (3)
\]

The policy would then record that \( a \) should be applied in any state \( S \) where \( S \models ps \). In effect, \( ps \) constrains \( S \) in a way that ensures action \( a \) and subsequent actions in the weak plan are applicable. To apply this same reasoning for the numeric case, we now derive an analogous constraint representation for numeric partial states.

Suppose in state \( S \), a variable \( v \) has value \( S[v] \) and variance \( S[\sigma^2(v)] \). Applying an action \( a \) with an uncertain numeric effect \( v + \mathcal{N}(p,q) \) (where \( p,q \in \mathbb{R} \)) reaches a state \( S' \), where:

\[
S'[v] = S[v] + p \quad S'[\sigma^2(v)] = S[\sigma^2(v)] + q \quad (4)
\]

If we have a precondition \( v \geq c \) that must be true in \( S' \), then \( S'[v] \geq c \) must be true with confidence \( \theta \) allowing for variance \( S'[\sigma^2(v)] \). By replacing \( S' \) with \( S \) according to (4), we can rewrite this as \( S[v] + p \geq c \) allowing for variance \( S[\sigma^2(v)] + q \). Writing the precondition in this form allows us to derive numeric regression for increase effects: for \( v \geq c \) to hold in the state after \( a \), then \( S[v] + p \geq c \) must hold allowing for variance \( S[\sigma^2(v)] + q \) in the state before \( a \). Each partial state \( ps \) in the policy records numeric constraints in this form. When checking if \( S \models ps \), these constraints

---

1 Department of Informatics, King’s College London, UK email: firstname.lastname@kcl.ac.uk
If a precondition allow it to prune fruitless search branches during policy-building. end is generalised by using the propositional RPG heuristic, to better are recorded each time a dead end is found. As noted earlier, the dead 3 Numeric Dead End Generalisation store the mean range of still-dead-end values for each variable. 2 where

\[ S[v] = k; \text{i.e., initially } S[v] \in [k, k]. \]

We use a variant in which the AUV (Boaty McBoatface) must satisfy a

REFERENCES


3 Numeric Dead End Generalisation

In PRP, when building a policy, forbidden state–action pairs (FSAPs) are recorded each time a dead end is found. As noted earlier, the dead end is generalised by using the propositional RPG heuristic, to better allow it to prune fruitless search branches during policy-building.

For the numeric case (with uncertainty), in a dead end state \( S \) we store the mean \( v \) and variance \( \sigma^2(v) \) of each variable. We generalise \( S \) by finding a range of values for each variable’s mean and variance such that \( S \) is still a dead end. This ties in nicely with the structure of the metric RPG heuristic [3], which underlies the heuristic we use in this work [5]. Within the metric RPG, the values of numeric variables are relaxed so they lie in a range rather than taking on fixed values. The upper and lower bounds on each value are then used to optimistically determine whether preconditions are true. Ordinarily, at the start of heuristic computation, the upper and lower bounds on each variable are set to the value the variable takes in the state being evaluated (in our case, the original dead end \( S \)). For generalising numeric dead ends as below, we exploit this mechanic to expand the range of still-dead-end values for each variable.

Suppose \( S \) is a dead end, and \( S[v] = k; \text{i.e., initially } S[v] \in [k, k]. \)

Tests comparing our approach (X axis) to a baseline (Y axis). For the baseline in all cases, only propositional dead end generalisation is used. For the to Solve baseline (a), additionally, only propositional regression is used. Tests that timed out are plotted at \( f = 10000 \). All axes are log scaled.

Figure 1. Tests comparing our approach (X axis) to a baseline (Y axis). For the baseline in all cases, only propositional dead end generalisation is used. For 2 We use a variant in which the AUV (Boaty McBoatface) must satisfy a prescribed number of goals, rather than considering oversubscription.