Assessing and Generating Robust Plans with Partial Domain Models

Tuan A. Nguyen * and **Subbarao Kambhampati** * and **Minh B. Do** †
* Dept. of Computer Science & Engineering, Arizona State University. Email: {natuan,rao}@asu.edu
† Embedded Reasoning Area, Palo Alto Research Center. Email: minh.do@parc.com
(For an updated version, please see: http://rakaposhi.eas.asu.edu/robust-plans.pdf)

Abstract

Most current planners assume complete domain models and focus on generating plans that are correct with respect to them. Unfortunately, assuming model completeness is unrealistic in the real world, where domain modeling remains a hard, labor-intensive and error-prone task. While domain experts cannot guarantee completeness, often they are able to circumscribe the incompleteness of the model by providing annotations as to which parts of the domain model may be incomplete. In such cases, the goal of planning would be to generate plans that are robust with respect to any known incompleteness of the domain. Doing this requires both a formal framework for assessing plan robustness and a methodology for guiding a planner's search towards robust plans. In this paper we formalize the notion of plan robustness with respect to a partial domain model, show a way of reducing exact robustness assessment to model-counting, and describe methods of approximately assessing plan robustness. We propose a heuristic search approach using model-counting techniques on top of the FF planner to generate plans that are not only correct but also robust, and present experimental results showing the effectiveness of this approach.

Introduction

In the past several years, significant strides have been made in scaling up plan synthesis techniques. We now have technology to routinely generate plans with hundreds of actions. A significant amount of ongoing work in the community has been directed at building up on these advances to provide efficient synthesis techniques under a variety of more expressive conditions (including partial observability, stochastic dynamics, durative/temporal actions, over-subscribed resources etc.).

All this work however makes a crucial assumption–that a complete model of the domain is specified in advance. While there may be domains where knowledge-engineering such detailed models is necessary as well as feasible (e.g., mission planning domains in NASA and factory-floor planning), there are also many scenarios where insistence on correct and complete models renders the current planning technology unusable. What is needed to handle such cases is a planning technology that can get by with partially complete domain models, and yet generate plans that are "robust" in the sense that they are likely to execute successfully in the real world.

This paper addresses the problem of assessing plan robustness and generating robust plans given partially complete domain models. Following (Garland and Lesh 2002), we shall assume that although the domain modelers cannot provide complete models, often they are able to provide annotations on the partial model circumscribing the places where it is incomplete. In our framework, these annotations consist of allowing actions to have *possible* pre-conditions and effects (in addition to the standard necessarily preconditions and effects).

As an example, consider a variation of the *Gripper* domain, a well-known planning benchmark. The robot has two hands that can be used to pickup balls from one room and move them to another room. The modeler suspects that one arm may have an internal problem, but this cannot be confirmed until the robot actually executes the plan. If the arm has a problem, the execution of the *pick-up* action never succeeds, if the arm has no problem, it can always pickup a ball. The modeler can express his partial knowledge about the domain by annotating the pickup action with statement representing this possible effect.

We shall see that partial domain models with such possible effects/preconditions implicitly defines an exponential set of complete domain models (each corresponding to particular realizations of possible preconditions/effects), with the semantics that the real domain model is guaranteed to be one of these. The robustness of a plan can now be formalized in terms of the fraction of the complete domain models under which it executes successfully. We shall show that robustness defined this way can be compiled into a (weighted) model-counting problem that works off of a causal-proof based SAT encoding of the plan (c.f. Mali and Kambhampati 1999). To generate robust plans, we extend the FF planner and combine its forward heuristic search with a SAT model-counting software. We present experimental results showing that the modified planner finds more robust plans than the base planner that ignores the incompleteness annotations.

Before we go further, we should clarify that the semantics of the possible preconditions/effects in our partial domain models differ fundamentally from non-deterministic and stochastic effects. Going back to the *Gripper* example above, with non-determinism/stochasticity, each time a robot tries to pickup a ball with the same hand, it will either succeed or fail independently of the other execution instances of the same action. In planning with incomplete action models, executing different instances of the same pick-

up action would either all fail or all succeed (because there is no uncertainty but the information is unknown at the time the model is built). This distinction also leads to differences between "robust plans" and good stochastic plans. If there is only one hand and the modeler does not know if it's good, then the most robust plan is to use that hand, but the robustness of the plan is measured by 50% success-rate, *no matter how many instances of that action we concatenate into the plan*. In contrast, if the hand's effects are stochastic, then trying the same picking action multiple times increases the chances of success.

Partial Domain Models

We consider planning scenarios where a planner is given as input a deterministic domain model $\mathcal{D}$ and a planning problem $\mathcal{P}$, together with some knowledge about the limited completeness of some actions specified in $\mathcal{D}$. Due to the presence of such knowledge, the partial domain model $\mathcal{D}$ should be seen as a stand in for (a possibly exponential number of) complete domain models which subsume the true domain model $\mathcal{D}^*$. A plan π that solves the problem $\mathcal{P}$ with respect to $\mathcal{D}$ thus may or may not succeed when executed (as it may not be correct with respect to $\mathcal{D}^*$. Since any of the completions of the partial domain model may correspond to the true domain model, the robustness of a plan can thus be defined as the fraction of completions in which it succeeds. We formalize these notions below:

A *partial domain model* $\mathcal{D}$ is defined as $\mathcal{D} = \langle F, A \rangle$, where $F = \{p_1, p_2, ..., p_n\}$ is the set of *propositions*, A is the set of *actions* that might *not* be completely specified. A *state* $s \subseteq F$ is set of propositions that hold in the state of the world, and the truth value of a proposition p in state s is denoted by $s[p]$. In addition to proposition sets that are known as its preconditions $Pre(a) \subseteq F$, additive effects $Add(a) \subseteq F$ and delete effects $Del(a) \subseteq F$, each action $a \in A$ in our formalism [1] is also modeled with the following sets of propositions:

- Possible precondition set $PreP(a) \subseteq F$ contains propositions that action a *might* need as its precondition.
- Possible additive effect set $AddP(a) \subseteq F$ contains propositions that action a *might* add after its execution.
- Possible delete effect set $DelP(a) \subseteq F$ contains propositions that action a *might* delete after its execution.

In addition, each possible precondition, additive and delete effect p of the action a are associated with a weight $w_a^{pre}(p)$, $w_a^{add}(p)$ and $w_a^{del}(p)$ ($0 \leq w_a^{pre}(p), w_a^{add}(p), w_a^{del}(p) \leq 1$) representing the domain writer's assessment of the likelihood that p is a precondition, additive and delete effect of a (respectively). Our formalism therefore allows the modeler to express her degree of belief on the likelihood that various possible preconditions/effects will actually be realized in the real domain model, and possible preconditions and effects without associated weights are assumed to be governed by non-deterministic uncertainty.

[1] We extend the formalism introduced by Garland & Lesh (2002), discussed in more details in the related work section.

The action a is considered incompletely modeled if either its possible precondition or effect set is non-empty. The action a is *applicable* in a state s if $Pre(a) \subseteq s$, and the resulting state is defined by $\gamma(s, a) = (s \setminus Del(a) \cup Add(a) \cup AddP(a))$.[2]

A *planning problem* is $\mathcal{P} = \langle \mathcal{D}, I, G \rangle$ where $I \subseteq F$ is the set of propositions that are *true* in the *initial state*, and G is the set of *goal propositions*. We denote $a_I, a_G \notin A$ as two dummy actions representing the initial and goal state such that $Pre(a_I) = \emptyset$, $Add(a_I) = I$, $Pre(a_G) = G$, $Add(a_G) = \{\top\}$ (where $\top \notin F$ denotes a dummy proposition representing goal achievement). A *plan* for the problem $\mathcal{P}$ is a sequence of actions $\pi = (a_0, a_1, ..., a_n)$ with $a_0 \equiv a_I$ and $a_n \equiv a_G$ and a_i is applicable in the state $s_i = \gamma(...\gamma(\gamma(a_0, \emptyset), a_1), ..., a_{i-1})$ $(1 \leq i \leq n)$.

In the presence of $PreP(a)$, $AddP(a)$ and $DelP(a)$, the execution of a plan π might not reach a goal state (i.e. the plan fails) when some possible precondition or effect of an action a is *realized* (i.e. winds up holding in the true domain model) and invalidates the executability of the plan.

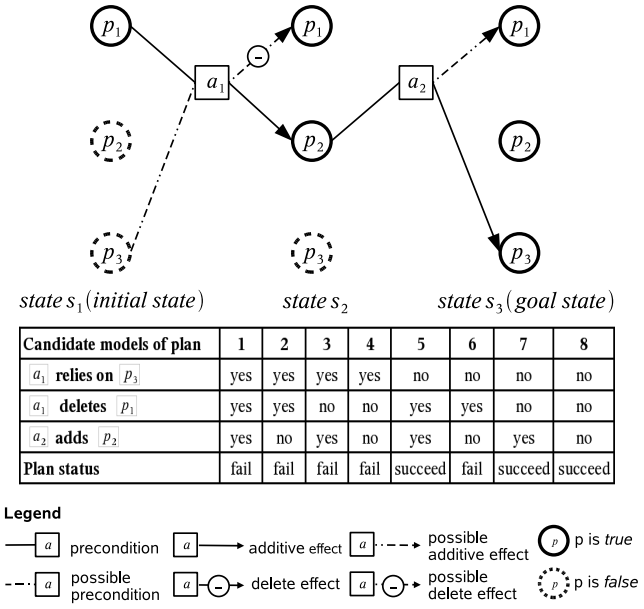
From the modeling point of view, the possible precondition and effect sets can be modeled at either the schema or grounded level. Thus, they can be relevant at the ground action level or at the general action schema level (and thus applicable to all ground actions sharing the same action schema). Going back to the *Gripper* domain mentioned earlier, if the possible internal problem is related to only the left hand, then only the ground action related to the left robot hand is incomplete. However, if the problem is related to both hands of the robot, then the incompleteness is at the action schema that models picking up using any hand.

Assumption underlying our model: In using $PreP$, $AddP$ and $DelP$ annotations, we are using an assumption, which we call *uncorrelated incompleteness*: the incomplete preconditions and effects are all assumed to be independent of each other. Our representation thus does not allow a domain writer to state that a particular action a will have the possible additive effect e only when it has the possible precondition p. While we cannot completely rule out a domain modeler capable of making annotations about such correlated sources of incompleteness, we assume that this is less likely.

Robustness Measure of Plans

Using the partial domain model as defined above, we can now formalize the notion of plan robustness. Given that any subset of possible precondition and effect sets of an action $a \in A$ can be part of its preconditions and effects in the complete domain $\mathcal{D}^*$, there are (exponentially) large number of *candidate* complete models for $\mathcal{D}^*$. For each of these candidate models, a plan $\pi = (a_0, a_1, ..., a_n)$ that is found in a plan generation process with respect to the partial domain model $\mathcal{D}$, as defined above, may either succeed to reach a goal state or fail when one of its actions, including a_g, cannot execute. If all of them have equal chance to be the complete model (according to the modeler), then the plan π is

[2] Note that we neglect $PreP(a)$ in action applicability checking condition and $DelP(a)$ in creating the resulting state to ensure the completeness. Thus, if there is a plan that is executable in at least one candidate domain model, then it is not excluded.



Candidate models of plan	1	2	3	4	5	6	7	8
a_1 relies on p_3	yes	yes	yes	yes	no	no	no	no
a_1 deletes p_1	yes	yes	no	no	yes	yes	no	no
a_2 adds p_2	yes	no	yes	no	yes	no	yes	no
Plan status	fail	fail	fail	fail	succeed	fail	succeed	succeed

Figure 1: An example of different candidate models of actions in plan, and the corresponding plan status. Spare nodes are actions, circle nodes with solid (dash) boundary are propositions that are *true* (*false*). The solid (dash) lines represent the (possible) preconditions and effects of actions.

considered highly *robust* if there are a large number of candidate models of $\mathcal{D}^*$ for which the execution of π successfully achieves all goals.

Example: Figure 1 shows an example with the partial domain model $\mathcal{D} = \langle F, A\rangle$ with $F = \{p_1, p_2, p_3\}$ and $A = \{a_1, a_2\}$ and a solution plan $\pi = (a_1, a_2)$ for the problem $\mathcal{P} = \langle \mathcal{D}, I = \{p_1\}, G = \{p_1, p_2, p_3\}\rangle$. All actions are incompletely modeled: $Pre(a_1) = \{p_1\}$, $PreP(a_1) = \{p_3\}$, $Add(a_1) = \{p_2\}$, $AddP(a_1) = \emptyset$, $Del(a_1) = \emptyset$, $DelP(a_1) = \{p_1\}$; $Pre(a_2) = \{p_2\}$, $PreP(a_2) = \emptyset$, $Add(a_2) = \{p_3\}$, $AddP(a_2) = \{p_1\}$, $Del(a_2) = DelP(a_2) = \emptyset$. Given that the total number of possible preconditions and effects is 3, the total number of candidate models of D^* therefore is $2^3 = 8$, for each of which it may succeed or fail to reach a goal state, as shown in the table. The candidate model 6, for instance, corresponds to the scenario where the first action a_1 does not depend on p_3 but it deletes p_1. Even though a_2 could execute, it does not have p_1 as an additive effect, and the plan fails to achieve p_1 as a goal. In summary, there are 5 of 8 candidate models where π fails and 3 candidate models of $\mathcal{D}^*$ for which π succeeds.

We define the *robustness* measure of a plan π, denoted by $R(\pi)$, as the probability that it succeeds in achieving goals with respect to $\mathcal{D}^*$ after execution. More formally, let $K = \sum_{a\in A}(|PreP(a)| + |AddP(a)| + |DelP(a)|)$, $S_D = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_{2^K}\}$ be the set of the candidate models of $\mathcal{D}^*$ and $h : S_\mathcal{D} \rightarrow [0, 1]$ be the distribution function (such that $\sum_{1\le i\le 2^K} h(\mathcal{D}_i) = 1$) representing the modeler's estimate of the probability that a given model in $S_\mathcal{D}$ is actually $\mathcal{D}^*$, the robustness value of a plan π is then defined as follows:

$$R(\pi) \stackrel{def}{=} \sum_{\mathcal{D}_j\in\prod} h(\mathcal{D}_j) \quad (1)$$

where $\prod \subseteq S_\mathcal{D}$ is the set of candidate models in which π is a valid plan.

Note that given the assumption of uncorrelated incompleteness, the probability $h(\mathcal{D}_i)$ for a model $\mathcal{D}_i \in S_\mathcal{D}$ can be computed as the product of the weights $w_a^{pre}(p)$, $w_a^{add}(p)$, and $w_a^{del}(p)$ (for all $a \in A$ and its possible preconditions/effects p) if p *is* realized as its precondition, additive and delete effect in $\mathcal{D}_i$ (or the product of their "complement" $1 - w_a^{pre}(p)$, $1 - w_a^{add}(p)$, and $1 - w_a^{del}(p)$ if p is *not*).

There is a very exetreme scenario, which we call *non-deterministic incompleteness*, when the domain writer does not have any quantitative measure of likelihood as to whether each (independent) possible precondition/effect will be realized or not. In this case, we will handle non-deterministic uncertainty as "uniform" distribution over models.[3] The robustness of π can then be computed as follows:

$$R(\pi) = \frac{|\prod|}{2^K} \quad (2)$$

The robustness value of the plan in Figure 1, for instance, is $R(\pi) = \frac{3}{8}$ if h is the uniform distribution. However, if the writer thinks that p_3 is very unlikely to be a precondition of a_1 (by providing $w_{a_1}^{pre}(p_3) = 0.1$), the robustness of this plan is increased to $R(\pi) = 0.9 \times 0.5 \times 0.5 + 0.9 \times 0.5 \times 0.5 + 0.9 \times 0.5 \times 0.5 = 0.675$ (as intutively, the first 4 candidate models with which the plan fails are very unlikely to be the complete one). For the rest of this paper, we first focus on the situation with the assumption of non-deterministic incompleteness, but some of our techniques can be adapted for the more general case.

Assessing Plan Robustness

Given the partial domain model $\mathcal{D}$ and a plan $\pi = (a_0, ..., a_n)$, computing K is easy and thus the main task of assessing plan robustness is to compute $M_\pi = |\prod|$ as accurately and as quickly as possible. A naive approach is to enumerate all domain models $\mathcal{D}_i \in S_\mathcal{D}$ and check for executability of π with respect to $\mathcal{D}_i$. This is prohibitively expensive when K is large. In this section, we introduce different approaches (from exact to approximate computation) to assessing the robustness value of plans.

Exact Computation Approach

In this approach, we setup a SAT encoding E such that there is a one-to-one map between each model of E with a candidate domain model $\mathcal{D} \in \prod$. Thus, counting the number of models of E should gives us M_π. To encode the executability of π, we will use constraints representing the *causal-proof* (c.f. Mali and Kambhampati 1999) of the correctness of π. In essence, the SAT constraints enforce that whenever an action $a_i \in \pi$ needs a precondition $p \in Pre(a_i)$ or

[3] as is typically done when distributional information is not available–since uniform distribution has the highest entropy and thus makes least amount of assumptions.

$p \in PreP(a_i)$, then (i) p is established at some level $j \leq i$ and (ii) there is no action that deletes p between j and i. In level i, we means the i^{th} state progressed from the initial state I by applying the action sequence $a_1, ...a_i$. The details of our compilation approach are given below:

SAT Boolean Variables: For each action $a \in A$, we create a set of boolean variables whose truth values represent whether a depends on, adds or deletes a proposition in its possible precondition and effect lists. Specifically, for each proposition $f \in PreP(a)$, we create a boolean variable f_a^{pre} where $f_a^{pre} = \mathbf{T}$ if f is realized as a precondition of a during execution, and $f_a^{pre} = \mathbf{F}$ otherwise. Similarly, we create boolean variables f_a^{add} and f_a^{del} for each $f \in AddP(a)$ and $f \in DelP(a)$. Note that these variables are created independently of solution plans and there are exactly 2^K possible complete assignments to all of these variables (with K defined in the previous section). Each complete assignment is a potential solution of E and corresponds to a candidate domain model of $\mathcal{D}^*$.

As mentioned above, it is possible that different actions $a_{i_1}, a_{i_2}, ..., a_{i_r}$ are grounded from the same action schema. In this case, if the incompleteness is specified at the schema level, then boolean variables created from a possible precondition (or effect) f of these actions are treated as a *single* variable, or $f_{a_{i_1}}^{pre} \equiv f_{a_{i_2}}^{pre} \equiv ... \equiv f_{a_{i_r}}^{pre}$.

SAT Constraints: As mentioned above, we encode the causal-proof of plan correctness in which all action precondition should be achieved at some level before it is needed. In order to identify correctly the set of actions whose (possible) effects can affect fact f needed at level i, we first introduce the notion of *confirmed level* C_f^i. Basically, C_f^i is the latest level at which the value of f is confirmed (to be either $\mathbf{T}$ or $\mathbf{F}$) by action a_j with $j < i$. Formally speaking: $f \in Pre(a_j)$ or $f \in Add(a_j)$ (confirmed $\mathbf{T}$) or $f \in Del(a_j)$ (confirmed $\mathbf{F}$). For example, $C_{p_2}^3 = 1$ and $C_{p_3}^2 = 0$ for the plan in Figure 1. Given that the initial state I at level 1 is a complete state, C_f^i exists for all f and i. Intuitively, if a fact f is needed at level i, then only the possible effects of actions executing within $[C_f^i, i-1]$ can affect the value of f at i.

Precondition establishment: For each $f \in Pre(a_i)$, we create the following constraints:

$$(C1) \quad \forall k \in [C_f^i, i-1] : \; f_{a_k}^{del} \Rightarrow \bigvee_{k<m<i} f_{a_m}^{add}$$

The constraint *C1* ensures that if f is a precondition of the action a_i and is deleted by a possible effect of a_k, then there must be another (white-knight) action a_m that re-establishes f as part of its possible add effect. Note that if there is no such action a_m that can add f, then we replace *C1* with $f_{a_k}^{del} \Rightarrow \mathbf{F}$.

If f is confirmed to be $\mathbf{F}$ at C_f^i, then we also need to add another constraint to ensure that it is added before i:

$$(C2) \quad \forall k \in [C_f^i, i-1] : \bigvee f_{a_k}^{add}$$

Note that based on our definitions of action application and valid plan, there should exist at least one such action a_k when setting up *C2*.

Possible Precondition Establishment: When a possible precondition is realized to be a true precondition, it also needs to be established as a normal precondition. Thus, if for a given action a_i and its possible precondition $f \in PreP(a_i)$, in a model where f is realized (i.e., $f_{a_i}^{pre} = \mathbf{T}$), we need to establish it and protect the establishment using constraints *C1* and *C2* above. The modification to *C1* and *C2* is that they are enforced only when $f_{a_i}^{pre} = \mathbf{T}$. Specifically, we add:

$$(C3) \quad \forall k \in [C_f^i, i-1] : \; f_{a_i}^{pre} \Rightarrow (f_{a_k}^{del} \Rightarrow \bigvee_{k<m<i} f_{a_m}^{add})$$

which is a variation of *C1*. If there is no such a_k action where $f \in DelP(a_k)$, then *C3* changes to: $f_{a_i}^{pre} \Rightarrow \mathbf{T}$ (i.e. we can omit the constraint). On the other hand, if for a given action a_k that may delete f and there is no action a_m to possibly add it after k, then we change *C3* to: $f_{a_i}^{pre} \Rightarrow (f_{a_k}^{del} \Rightarrow \mathbf{F})$.

When f is confirmed to be $\mathbf{F}$ at C_f^i, then we also add a variation of *C2*:

$$(C4) \quad \forall k \in [C_f^i, i-1] : \; f_{a_i}^{pre} \Rightarrow \bigvee f_{a_k}^{add}$$

if there is no such action a_k that can possibly add f, then constraint *C4* changes to $f_{a_i}^{pre} \Rightarrow \mathbf{F}$.

Therefore, given a logical formula representing causal-proof for plan correctness, an exact model-count method such as *Cachet* (Sang et al. 2004) can be invoked to compute the number of models with which the plan succeeds.

Using precondition/effect weights in computing robustness: in order to relax the assumption of non-determinstic incompleteness, the SAT boolean variables f_a^{pre}, f_a^{add}, f_a^{del} can be associated with corresponding weights $w_a^{pre}(f)$, $w_a^{add}(f)$, $w_a^{del}(f)$ if provided, and a model-counting algorithm (for instance, the one described in (Sang, Beame, and Kautz 2005)) can be used to compute the weight of the logical formula, which immediately corresponds to the plan robustness.

Approximate Computation Approaches

The exact computation discussed above has exponential running time in the worst case. In some cases, it is enough to know approximate robustness values of plans, for instance in comparing two plans whose robustness are very different. We now discuss two approximate approaches to estimate plan robustness.

Using approximate model-counting algorithms: Given a logical formula representing constraints on domain models with which the plan π succeeds as described in the previous section, in this approach an approximate model-counting software, for instance the work by (Gomes, Sabharwal, and Selman 2006) or (Wei and Selman 2005), can be used as a black-box to approximate the number of domain models.

Robustness propagation approach: We next consider an approach based on approximating robustness value of each action in the plan, which can then be used later in generating robust plans. At each action step i $(0 \leq i \leq n)$, we denote $M_\pi(i) \subseteq S_\mathcal{D}$ as the number of domain models with which all actions $a_0, a_1, ..., a_i$ of π are executable, and therefore

$R_\pi(i) = |M_\pi(i)|/2^K$ as the robustness value of the action step i. Similarly, we define the robustness value for any set of propositions $Q \subseteq F$ at level i, $R_\pi(Q, i)$ $(i > 0)$, as the ratio of $S_\mathcal{D}$ with which $(a_0, a_1, ..., a_{i-1})$ succeeds and $p = \mathbf{T}$ $(\forall p \in Q)$ in the resulting state s_i.

The purpose of this approach is to estimate the robustness values $R_\pi(i)$ through a propagation procedure, starting from the dummy action a_0 with a note that $R_\pi(0) = 1$. The resulting robustness value $R_\pi(n)$ at the last action step can then be considered as an approximate robustness value of π. Inside the propagation procedure (Algorithm 1) is a sequence of approximation steps: at each step i $(1 \leq i \leq n)$, we estimate the robustness values of individual propositions $p \in F$ and of the action a_i (the procedure **ApproxPro(**p, i, π**)** at lines 6-7 and **ApproxAct(**i, π**)** at line 8, respectively) using those of the propositions and action at the previous step. To obtain efficient computation, in the following discussion we assume that the robustness value of a proposition set $Q \subseteq F$ can be approximated by a combination of robustness values of $p \in Q$, and that the precondition and effect realizations of different actions are independent (although two actions can be instantiated from an action schema, and incompleteness information is asserted at the schema level).

The procedure **ApproxPro(**p, i, π**)** is presented in the algorithm 2. When $p \notin s_i$ (this includes the case where $p \in Del(a_{i-1})$), the robustness of p is set to 0 (line 4-5), since $p = \mathbf{F}$ at its confirmed level C_p^i and yet cannot be (possibly) added by any action a_k $(C_p^i \leq k \leq i-1)$. Now we consider the case when $p \in s_i$. If $p \in Add(a_{i-1})$, then p will be asserted in the state s_i by a_{i-1} for all domain models with which $(a_0, a_1, ..., a_{i-1})$ succeeds, and hence its robustness is that of a_{i-1} (line 6-7). When p is a possible additive effect of a_{i-1} (line 8-13), we consider two cases:

- $p \notin s_{i-1}$: $p = \mathbf{T}$ in the resulting state s_i after $(a_0, a_1, ..., a_{i-1})$ executes only for candidate domain models with which (i) $(a_0, a_1, ..., a_{i-1})$ succeeds and (ii) p is realized as additive effect of a_{i-1}. With the independence assumption on precondition and effect realizations of actions, the number of such models is $\frac{1}{2} \times |M_\pi(i-1)|$, therefore the robustness of p in the state s_i is approximated with $\frac{1}{2} \times R_\pi(i-1)$ (line 9-10).
- $p \in s_{i-1}$: $p = \mathbf{T}$ in s_i with the following two *disjoint* sets of candidate domain models:
  - Those models with which (i) $(a_0, a_1, ..., a_{i-1})$ succeeds, and (ii) p *is* additive effect of a_{i-1}. Similar to the case when $p \notin s_{i-1}$, the number of such models is approximated with $\frac{1}{2} \times |M_\pi(i-1)|$.
  - Those models with which $(a_0, a_1, ..., a_{i-1})$ succeeds, p is *not* additive effect of a_{i-1}, and $p = \mathbf{T}$ in s_{i-1} during execution. Equivalently, they are the models with which (i) p and all propositions in $S \cup Pre(a_{i-1})$ are $\mathbf{T}$ at the step $i-1$ (for any $S \subseteq PreP(a_{i-1})$)—there are $R_\pi(\{p\} \cup S \cup Pre(a_{i-1}), i-1) \times 2^K$ such models, and (ii) p is not additive effect of a_{i-1}, all propositions in S are realized as preconditions of a_{i-1}. As these realizations are assumed to be independent on actions at levels before $i-1$, the number of such models is approximated with: $\sum_{S \subseteq PreP(a_{i-1})} \frac{1}{2} \times \frac{1}{2^{|PreP(a_{i-1})|}} \times R_\pi(\{p\} \cup S \cup Pre(a_{i-1}), i-1) \times 2^K$.

  These approximate numbers of models in these two sets are then used to estimate the robustness value of p (line 11-12).

With similar arguments, we approximate the robustness of p in the state s_i when $p \in s_i$, p is possible delete effect of a_{i-1} (line 14-16) and when $p \in s_i$ but is neither possible precondition nor possible effect of a_{i-1} (line 17).

The procedure **ApproxAct(**i, π**)** (Algorithm 3) approximates the robustness of the action a_i, using the robustness values of its (possible) sets of preconditions $S \cup Pre(a_i)$ $(S \subseteq PreP(a_i))$. Again, with the realization independence assumption, for any set of realized preconditions of a_i, $S \subseteq PreP(a_i)$, here we approximate the number of domain models for which a_i is executable with $\frac{1}{2^{|PreP(a_i)|}} \times R_\pi(S \cup Pre(a_i), i)$.

Until now, we haven't mentioned in our algorithms any specific way to approximate the robustness value of a proposition set $Q \subseteq F$ from that of its individual propositions $p \in Q$. We discuss two possible options:

- One simple method is to assume that any two different propositions $p, q \in Q$ are independent, in other words the sets of domain models with which they are made $\mathbf{T}$ at the step i are drawn independently, and therefore $R_\pi(Q, i) = \prod_{p \in Q} R_\pi(\{p\}, i)$.
- The interaction between propositions (Bryce and Smith 2006) can also be considered to have better approximate value of $R_\pi(Q, i)$. The interaction degree of proposition pairs is now propagated through the actions of π (instead of the plan graph), starting from the initial state where all propositions are known to be independent, which is used later together with robustness value of individual propositions $p \in Q$ in order to estimate $R_\pi(Q, i)$.

Figure 2 shows an example of the robustness propagation procedure, assuming that the robustness of a proposition set is approximated with the product of the robustness of its propositions. The robustness value of action step 2, for instance, is computed by considering two cases: when p_1 is realized as a precondition of a_2, and when it is not. Therefore, $R_\pi(2) = \frac{1}{2} \times R_\pi(\{p_1, p_3\}, 2) + \frac{1}{2} \times R_\pi(\{p_3\}, 2) = \frac{1}{2} \times (0.5 \times 0.5) + \frac{1}{2} \times 0.5 = 0.375$. On the other hand, $p_1 = \mathbf{T}$ at step 3 if it is realized as precondition of a_2, or it must be $\mathbf{T}$ at level 2 and a_2 is executable. Hence, $R_\pi(\{p_1\}, 3) = \frac{1}{2} \times R_\pi(2) + \frac{1}{2} \times \frac{1}{2} \times (R_\pi(\{p_1\} \cup \{p_3\}, 2) + R_\pi(\{p_1\} \cup \{p_1, p_3\}, 2)) = \frac{1}{2} \times 0.375 + \frac{1}{2} \times \frac{1}{2} \times (0.25 + 0.25) = 0.3125$. Note that if there is an action a_3 applying in the state s_3, and the precondition and effect realizations of two actions a_1, a_3 must be consistent in any domain models, our robustness propagation treats them as two independent actions so that the robustness of a_3 at level 3 can be approximated using robustness of propositions and the action at level 2 only (and "forgetting" the action a_1).

Generating Robust Plans

Given the robustness measure defined previously, in this section we introduce our first attempt to develop a forward-search approach to generating robust plans. To this end, we

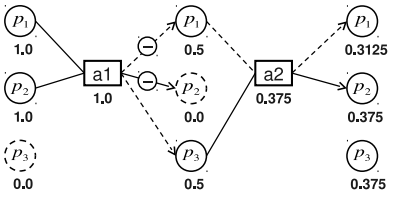


Figure 2: An example of robustness propagation.

Algorithm 1: Approximate plan robustness.

Input: The plan $\pi = (a_0, a_1, ..., a_n)$;
Output: The approximate robustness value of π;
begin
  $R_\pi(0) = 1$;
  for $i = 1..n$ **do**
    for $p \in F$ **do**
      $R_\pi(\{p\}, i) \leftarrow$ ApproxPro(p, i, π);
    $R_\pi(i) \leftarrow$ ApproxAct(i, π);
  Return $R_\pi(n)$;
end

extend the FF planner (Hoffmann and Nebel 2001) by incorporating an exact model-counting procedure into its *enforced hill climbing* strategy to assess the robustness value of both the current partial plan and relaxed plan, which then is used together with the heuristic value in chosing a better state.

Given a state s reached from the initial state s_1 through a sequence of actions $\pi(s)$, and the corresponding relaxed plan $RP(s)$, we compute the number of candidate domain models for which all actions in $\pi(s)$ are executable and $RP(s)$ remains valid relaxed plan by enforcing a set of constraints as follows. The set of constraints on the sequence of actions $\pi(s)$, called $C_\pi(s)$, is constructed using the constraints $C1$ - $C4$ above. The constraint set $C_{RP}(s)$ for $RP(s)$, on the other hand, is put together differently, respecting the fact that it may become *invalid* in the complete domain $\mathcal{D}^*$: some realized precondition $f \in PreP(a)$ of the action $a \in A_s(t)$ at the level t of the relaxed plan $RP(s)$ may no longer be supported by the (realized) additive effects of any action in the relaxed plan at the previous levels. A relaxed plan is therefore considered more robust if it remains valid in a larger number of candidate domain models of $\mathcal{D}^*$. For each action $a \in A_s(t)$ and $f \in PreP(a)$ such that f is not an additive effect of any action at levels before t, we add the following constraint into $C_{RP}(s)$:

$$(C5) \quad f_a^{pre} \Rightarrow \bigvee_{a' \in AddP_{RP}^{-1}(f,t)} f_{a'}^{add}$$

where $AddP_{RP}^{-1}(f,t)$ is the set of actions of the relaxed plan at levels $t' < t$ having f as a possible additive effect. The union set of constraints $C_\pi(s) \cup C_{RP}(s)$ is then given to a model-counting software to get the number of models, from which the robustness value $r(s)$ is computed.

The procedure described aboved is applied during the enforced hill-climbing search for both the current state s_i and its descendant states s_{des} to assess the robustness values $r(s_i)$ and $r(s_{des})$. These are then used together with the

Algorithm 2: The procedure ApproxPro(p, i, π) to approximate the robustness value of a proposition.

Input: The plan $\pi = (a_0, a_1, ..., a_n)$; proposition $p \in F$; level i $(1 \leq i \leq n)$;
Output: The approximate robustness value of p at level i;
begin
  if $p \notin s_i$ **then**
    return 0;
  if $p \in Add(a_{i-1})$ **then**
    return $R_\pi(i-1)$;
  if $p \in AddP(a_{i-1})$ **then**
    if $p \notin s_{i-1}$ **then**
      $r \leftarrow \frac{1}{2} \times R_\pi(i-1)$;
    else
      $r \leftarrow \frac{1}{2} \times R_\pi(i-1) + \frac{1}{2} \times \frac{1}{2^{|PreP(a_{i-1})|}} \times \sum_{S \subseteq PreP(a_{i-1})} R_\pi(\{p\} \cup S \cup Pre(a_{i-1}), i-1)$;
    return r;
  if $p \in DelP(a_{i-1})$ **then**
    $r \leftarrow \frac{1}{2} \times \frac{1}{2^{|PreP(a_{i-1})|}} \times \sum_{S \subseteq PreP(a_{i-1})} R_\pi(\{p\} \cup S \cup Pre(a_{i-1}), i-1)$;
    return r;
  return $\frac{1}{2^{|PreP(a_{i-1})|}} \times \sum_{S \subseteq PreP(a_{i-1})} R_\pi(\{p\} \cup S \cup Pre(a_{i-1}), i-1)$;
end

Algorithm 3: The procedure ApproxAct(i, π) to approximate the robustness value of an action.

Input: The plan $\pi = (a_0, a_1, ..., a_n)$; level i $(1 \leq i \leq n)$;
Output: The approximate robustness value of action a_i;
begin
  return $\frac{1}{2^{|PreP(a_i)|}} \times \sum_{S \subseteq PreP(a_i)} R_\pi(S \cup Pre(a_i), i)$;
end

original FF's heuristic values $h(s)$ and $h(s_{des})$ in comparing the two states. Specifically, we use these values simply for breaking ties between two states s_i, s_{des} having *similar* heuristic values: if $|h(s_i) - h(s_{des})| \leq \delta$ then the state with higher robustness value is considered better; otherwise the better state is the one with better (i.e., smaller) $h(s)$ value.

Empirical Evaluation: Preliminary Results

We have implemented our proposed approaches to find robust plans as described in the previous sections. In order to compute the number of models of a set of constraints, we exploit the *Cachet* software (Sang et al. 2004).

Partial domain generation: In order to test our approach with the benchmark domains, we have built a program to generate a partial domain model from a deterministic one. To do this, we add N new propositions to each domain (assumed *false* at the initial state) and make each action schema incomplete with a probability $p_{incomplete}$. We first generate copies of each action schema a: $a_1, a_2, \ldots a_K$, (for instance, the copies $Fly_1, Fly_2, \ldots, Fly_K$ for the Fly action schema in the ZenoTravel domain), and then to make these actions incomplete we randomly add $\min(p_{pre} \times |Pre(a)|, N)$ new propositions into the possible precondition list of a with a probability $p_{pre} > 0$, and do the same for its possible

add and possible delete lists (using p_{add}, $|Add(a)|$ and p_{del}, $|Del(a)|$ respectively). Each action schema may also include new propositions (randomly selected) as its additive (delete) effects with a given probability p_{new_add} (p_{new_del}). This strategy ensures that the solution to problems in the new domain exists when it is solvable in the original domain, and a new plan with different robustness value can be generated from another plan by replacing actions in the same original schemas (e.g., Fly_1 and Fly_2).

Analysis: We tested our approaches with three domains in IPC-3: Rovers, Satellite and ZenoTravel. For each domain, we first make 4 copies of each action schema, adding 5 new propositions, and randomly generating 3 different partial domains using our generators with $p_{incomplete} = 1.0$, $p_{pre} = p_{add} = p_{del} = p_{new\_add} = p_{new\_del} = 0.5$. In order to test our *Exact Model-Counting* (EMC) approach, we set $\delta = 1$. We compare the performance of the EMC approach with the *base-line* approach, which runs the FF planner on the partial domains, ignoring possible preconditions and effects, in terms of the following main objectives: (1) the robustness value of solution plans returned, (2) the time taken by the search (including the time used by the model-counting algorithm, Cachet, in the EMC approach). The experiments were conducted using an Intel Core2 Duo 3.16GHz machine with 4Gb of RAM. For both approaches, we search for a solution plan within the 30-minute time limit for each problem.

Overall, we observe that the EMC approach is able to improve the robustness of solution plans, compared to the base-line FF. In particular, over all solvable problems of the 3 versions of each partial domain generated, the EMC approach returns more robust plans in 49/58 (84.48%) problems in Rovers domain, 33/54 (61.11%) in Satellite, and 35/60 (58.33%) in ZenoTravel; it returns less robust plans than FF in 6/58 (10.34%) problems of in Rovers, 1/54 (1.85%) in Satellite, and 7/60 (11.67%) in ZenoTravel; and the two approaches return the same plans in 13/54 (24.07%) problems in Satellite domain and 18/60 problems (30%) in ZenoTravel. However, the EMC approach fails to solve 3/58 (5.07%) problems in Rovers and 7/54 (12.96%) problems in Satellite.

Figure 3(a) shows the relative plan robustness value generated by the two approaches in *one* of the three partial versions of each domain. Among 19, 16 and 20 problems of Rovers, Satellite and ZenoTravel (respectively) that are solvable, EMC finds more robust plans than FF in 15/19 (78.94%) problems in Rovers domain, 9/16 (56.25%) problems in Satellite and 11/20 (55%) problems in ZenoTravel. The search of EMC approach, however, may lead to a solution plan with lower robustness value: 4/19 (21.05%) problems in Rovers domain; and as mentioned above it could also fail to find a solution: EMC fails in 4/16 (25%) problems in Satellite domain. One of the reasons for this somewhat counter-intuitive failure is that when EHC search in EMC approach, enhanced with model-counting procedure, fails to find a solution, there is not enough time remaining for the standard best-first search (used also in base-line FF approach) to find a solution. We also observe that if a plan is found in this case, its quality is normally not better than the one returned by the EHC of the FF planner.

Figure 3(b) shows the time spent by the two approaches searching for plans. We observe that even though EMC can improve the quality of plans, it can take much longer search time in some problems where the total time to count the number of models is high. In many problems, on the other hands, the overall search time of the EMC approach is not too expensive, compared to the base-line, and the solution plan is also better in quality.

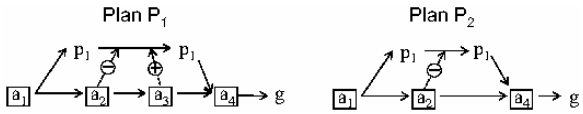


Figure 4: Example of plans with same risk, as defined in Garland & Lesh model, but with different robustness values.

Related Work

The work most closely related to ours is that of Garland & Lesh (2002). Although their paper shares the objective of generating robust plans with respect to partial domain models, their notion of robustness is defined in terms of four different types of risks, and only has tenuous heuristic connections with likelihood of successful execution of the plan (Figure 4). In contrast, we provide a more formal definition in terms of fraction of complete models under which the plan succeeds. A recent extension by Robertson & Bryce (2009) does focus on the plan generation in Garland & Lesh model, by measuring and propagating risks over FF's relaxed plans. Unfortunately, their approach still relies on the same unsatisfactory formulation of robustness. The work by Fox et al 2006 also explores robustness of plans, but their focus is on temporal plans and their executability under unforeseen execution-time variations. They focus only on assessing robustness, and do so with monte carlo probing techniques.

Although we focused on a direct approach for generating robust plans, it is also possible to compile the robust plan generation problem into a "conformant probabilistic planning" problem (Bryce, Kambhampati, and Smith 2008). Specifically, the realization of each possible precondition and effect can be seen as being governed by an unobservable random variable. This allows us to compile down model incompleteness into initial state uncertainty. Finding a plan with robustness p will then be equivalent to finding a probabilistic conformant plan P that reaches goals with a probability p. As of this writing, we do not have definitive conclusions on whether or not such a compilation method will be competitive with direct methods that we are investigating.

Our work can also be categorized as one particular instance of the general model-lite planning problem, as defined in (Kambhampati 2007). Kambhampati points out a large class of applications ranging from web-service to work-flow management where model-lite planning is unavoidable due to the difficulty in getting a complete model. While we focused on the problem of how to assess robustness and generate robust plans with the given partial domain model, in the long run, an agent should try to reduce the model incompleteness through learning. As such, the work on learning action models (e.g (Yang, Wu, and Jiang 2007; Amir and Chang 2008) is also relevant to the general problem we address.

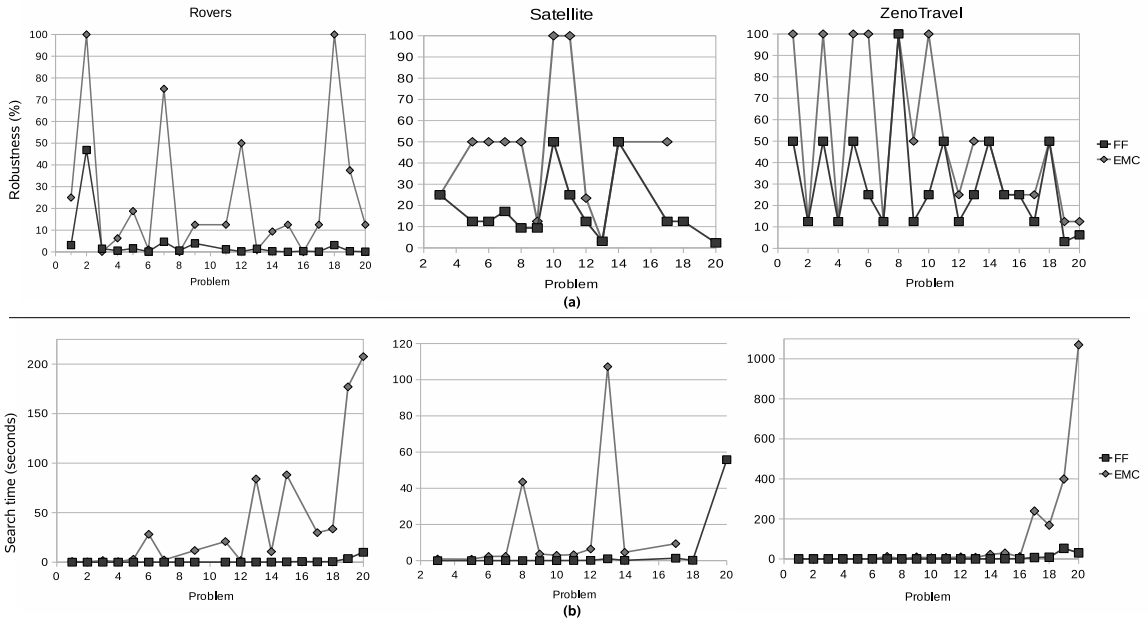


Figure 3: The robustness of plans and search time in *Rovers*, *Satellite* and *ZenoTravel*.

Conclusion and Future Work

In this paper, we motivated the need for assessing and generating robust plans in the presence of partially complete domain models. We presented a framework for representing partially complete domains models, where the domain modeler can circumscribe the incompleteness in the domain through possible precondition/effect annotations. We then developed a well-founded notion of plan robustness , showed how robustness assessment can be cast as a model-counting problem over a causal SAT encoding of the plan, and proposed an approximate approach based on robustness propagation idea. Finally, we described an approach for modifying FF so it can generate more robust plans. We presented empirical results showing the effectiveness of our approach.

We are investigating more sophisticated ways of assessing plan robustness, considering to some extent the consistent constraints between precondition and effect realizations of actions to have a better trade-off between robustness accuracy and computation cost. We are also extending our plan generation method to make its search more sensitive to robustness, taking into account the robustness values of actions estimated from the approximate robustness assessment approach. We intend to do this by modifying the relaxed plan extraction routine in FF so it is greedily biased towards more robust heuristic completions of the current partial plan.

Acknowledgement: This research is supported in part by ONR grants N00014-09-1-0017, N00014-07-1-1049 and the NSF grant IIS-0905672.

References

Amir, E., and Chang, A. 2008. Learning partially observable deterministic action models. *Journal of Artificial Intelligence Research* 33(1):349–402.

Bryce, D., and Smith, D. 2006. Using Interaction to Compute Better Probability Estimates in Plan Graphs. In *ICAPS'10 Workshop on Planning Under Uncertainty and Execution Control for Autonomous Systems*. Citeseer.

Bryce, D.; Kambhampati, S.; and Smith, D. 2008. Sequential monte carlo in reachability heuristics for probabilistic planning. *Artificial Intelligence* 172(6-7):685–715.

Fox, M.; Howey, R.; and Long, D. 2006. Exploration of the robustness of plans. In *AAAI*.

Garland, A., and Lesh, N. 2002. Plan evaluation with incomplete action descriptions. In *Proceedings of the National Conference on Artificial Intelligence*, 461–467.

Gomes, C.; Sabharwal, A.; and Selman, B. 2006. Model counting: A new strategy for obtaining good bounds. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, 54. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research (JAIR)* 14:253–302.

Kambhampati, S. 2007. Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain theories. In *AAAI*.

Mali, A., and Kambhampati, S. 1999. On the utility of plan-space (causal) encodings. In *AAAI*, 557–563.

Robertson, J., and Bryce, D. 2009. Reachability heuristics for planning in incomplete domains. In *ICAPS'09 Workshop on Heuristics for Domain Independent Planning*.

Sang, T.; Beame, P.; and Kautz, H. 2005. Solving Bayesian networks by weighted model counting. In *Proc. of AAAI-05*.

Sang, T.; Bacchus, F.; Beame, P.; Kautz, H.; and Pitassi, T. 2004. Combining component caching and clause learning for effective model counting. In *Seventh International Conference on Theory and Applications of Satisfiability Testing*. Citeseer.

Wei, W., and Selman, B. 2005. A new approach to model counting. *Lecture Notes in Computer Science* 3569:324–339.

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning action models from plan traces using weighted max-sat. *Artificial Intelligence Journal (AIJ)* 171:107–143.