Planning with Partial Preference and Domain Models

Tuan A. Nguyen
Department of Computer Science & Engineering
Arizona State University, Tempe, AZ 85287, USA
natuan@asu.edu
(Joint work with Subbarao Kambhampati, Minh Do and Biplav Srivastava.)

Introduction

This thesis aims to develop scablable plan synthesis techniques for scenarios where the models of a user's preferences and/or domain dynamics cannot be completely specified. As pointed out in (Kambhampati 2007), there is a wide range of applications such as web-service and workflow management in which it is very difficult to get a complete model, and therefore such "model-lite" planning methods become important.

While the solution concept to a planning problem is clearly understood when complete models of user's preferences and domain dynamics are available[1], it is not even obvious what the right solution to a planning problem should be if a partial model is given as input, let alone how to find it efficiently. Therefore, the contributions of my thesis are first to propose quality measures for solutions to this problem, and then to investigate various efficient techniques for generating high quality solutions in two cases of partial models. In particular,

- When the user's preference model is known to be incomplete, the planner's job changes from finding a single optimal plan to finding a set of *representative* solutions ("options") and present them to the user (in the hope that she will find one of them desirable). As a result, quality measures should be defined to evaluate plan sets with respect to the user's partial preferences. We therefore adapt the idea of *Integrated Preference Function* (IPF) (Carlyle et al. 2003) developed in Operations Research (OR) community in the context of multi-criteria scheduling to measure the expected utility that the user can get from the set.
- When the domain model is partially specified, a plan generated cannot be guaranteed to succeed during execution. All we can say is a plan with higher chance to achieve the goals should be considered better. In this case, we develop a *robustness* measure of plans estimating a portion of the space of possible complete domains for which the plan succeeds during execution (with respect to the *complete* model).

[1]In particular, it is the single *best* plan with respect to the user known preferences, and it is any *valid* plan which reaches a state satisfying all goals from an initial state given a complete domain model.

These quality measures, while providing clean definitions of solution concepts to the new planning scenarios, rise more challenges to plan synthesis techniques. The next contribution is to propose efficient methods to generate high quality solutions to planning problems under partially specified preferences and domain dynamics, working on top of the LPG (Gerevini, Saetti, and Serina 2003) and FF planners (Hoffmann and Nebel 2001). In the following, we first discuss the solution concept and plan synthesis techniques for planning with partial preference model, and then for situation where the domain model is incomplete. Finally, we end the paper with the conclusion.

Planning with Partial Preference Models

We first consider the problem of generating a set of representative plans in scenarios where the user has multiple (and possibly conflicting) plan objectives, but their relative importance degree cannot be completely specified. We concentrate on metric temporal planning where each action $a \in A$ has a duration d_a and execution cost c_a, and the user's preference model is formalized as follows:

- The desired objective function involves minimizing both components: the makespan of a plan p, $time(p)$, and its execution cost, $cost(p)$.
- The quality of a plan p is a convex combination: $f(p, w) = w \times time(p) + (1 - w) \times cost(p)$, where the weight $w \in [0, 1]$ represents the trade-off between the two competing objective functions.
- The belief distribution of w over the range $[0, 1]$ is known. If the user does not provide any information or we have not learned anything about the preference on the trade-off between *time* and *cost* of the plan, then the planner can assume a uniform distribution (and improve it later using techniques such as preference elicitation).

Given that the exact value of w is unknown, we cannot find a single optimal plan. The best strategy is therefore to find a representative set of non-dominated plans[2] minimizing the expected value of $f(p, w)$ with regard to the given distribution of w over $[0, 1]$.

[2]A plan p_1 is dominated by p_2 if $time(p_1) \geq time(p_2)$ and $cost(p_1) \geq cost(p_2)$ and at least one of the inequalities is strict.

Integrated Preference Function (IPF)

The *Integrated Preference Function* (IPF) (Carlyle et al. 2003) measure assumes that the user preference model is represented by two factors: (1) a probability distribution $h(\alpha)$ of parameter vector α such that $\int_\alpha h(\alpha)\, d\alpha = 1$ (in the absence of any special information about the distribution, $h(\alpha)$ can be assumed to be uniform), and (2) a function $f(p, \alpha) : \mathcal{S} \rightarrow \mathbb{R}$ (where $\mathcal{S}$ is the solution space) combines different objective functions into a single real-valued quality measure for solution p. The expected quality of a solution set $\mathcal{P} \subseteq \mathcal{S}$ is defined as:

$$IPF(\mathcal{P}) = \int_\alpha h(\alpha) f(p_\alpha, \alpha)\, d\alpha$$

where $p_\alpha = \underset{p \in \mathcal{P}}{\operatorname{argmin}}\, f(p, \alpha)$ is the best solution according to $f(p, \alpha)$ for each given α value. The set of plans with the minimal IPF value is most likely to contain the desired solutions that the user wants and in essense a good representative of $\mathcal{S}$. When f is convex combination of objective functions, as in our setting, the measure is called *Integrated Convex Preference* (ICP).

Finding Representative Plans Using ICP

We considered three different approximate techniques on top of the LPG planner (Gerevini, Saetti, and Serina 2003) to find a set $\mathcal{P}$ of at most k plans with a good ICP value.

1. **Sampling weight values:** Given the distribution $h(w)$ of trade-off value w is known, this approach first samples a set of k values for w: $\{w_1, w_2, ..., w_k\}$, and then for each w_i $(1 \leq i \leq k)$ search for a plan p_i minimizing the value of $f(p, w_i)$.
2. **ICP sequential approach:** In this approach, we incrementally built the plan set $\mathcal{P}$ by finding a plan p such that $\mathcal{P} \cup \{p\}$ has the lowest ICP value, starting with an empty solution set $\mathcal{P} = \emptyset$.
3. **Hybrid approach:** This approach aims to combine the strengths of both sampling and ICP sequential approaches. Specifically, we use sampling to find several plans optimizing for different weights, and these plans are then used to seed the subsequent ICP-sequential runs.

Our expriment suggests that a combination of sampling and ICP Sequential to exploit their individual benefits would be the best choice. This work has been presented in (Nguyen et al. 2009).

Planning with Partial Domain Models

We next consider planning scenarios where a planner is given as input a deterministic domain model $\mathcal{D} = (F, A)$ and a planning problem $\mathcal{P} = \langle \mathcal{D}, I, G \rangle$, together with some knowledge about the limited completeness of some actions specified in $\mathcal{D}$, where F is the set of propositions, A set of actions, $I \subseteq F$ an initial state, and G a set of goal propositions. As a variation of the formalism introduced in (Garland and Lesh 2002), each action $a \in A$ (in addition to its preconditions $Pre(a) \subseteq F$, additive effects $Add(a) \subseteq F$, delete effects $Del(a) \subseteq F$) 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 (delete) effect set $AddP(a) \subseteq F$ ($DelP(a) \subseteq F$) contains propositions that action a *might* add (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.

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 $\gamma(s, a) = (s \setminus Del(a) \cup Add(a) \cup AddP(a))$.[3] 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.

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

[3] 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.

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. The plan π therefore is 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.

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 $K = \sum_{a \in A}(|PreP(a)| + |AddP(a)| + |DelP(a)|)$, $S_\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_{2^K}\}$ be the set of the candidate models of $\mathcal{D}^*$ and $h : S_\mathcal{D} \rightarrow [0, 1]$ be the distribution function (with $\sum_{1 \leq i \leq 2^K} h(\mathcal{D}_i) = 1$) 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. 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 *nondeterministic 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.[4] The robustness of π can then be computed as follows:

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

Note that the partially specified model of domain dynamics causes uncertainty in plan executability in a different way with stochastic planning. A robot that plans to pick up a box with one hand, which is suspected to have an internal problem by the modeler and cannot be tested until plan execution, has 50% success-rate, *no matter how many instances of that action executed*. On the other hand, a *pick-up* action with 0.5 probability of success in stochastic planning can be tried for many times to increase the chance of success.

Assessing Plan Robustness

A naive approach to assessing plan robustness, as defined in (2), is to enumerate all domain models $\mathcal{D}_i \in S_\mathcal{D}$ and check for executability of π with respect to $\mathcal{D}_i$, which is prohibitively expensive when K is large. In this section, we first propose an exact computation method using model-counting technique that can be significantly faster (though the worst-case complexity does not change), and then discuss approximate approaches to assessing plan robustness.

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

Exact Computation Given a partial domain model $\mathcal{D}$ and a plan $\pi = (a_0, a_1, ..., a_n)$, we will setup the SAT encoding E representing the *causal-proof* (c.f. Mali and Kambhampati 1999) of the correctness of π such that there is a one-to-one map between each model of E with a candidate domain model $\mathcal{D} \in \prod$. The exact robustness value of π, therefore, can be computed by invoking any exact weighted model-counting software, as the one described in (Sang, Beame, and Kautz 2005), given E as an input. The compilation is briefly described as follows:

SAT boolean variables: For each action $a \in A$ and $f \in PreP(a)$, we create a boolean variable f_a^{pre} with a weight $w_a^{pre}(f)$ where $f_a^{pre} = \mathbf{T}$ (*true*) if f is realized as a precondition of a during execution, and $f_a^{pre} = \mathbf{F}$ (*false*) otherwise. Similarly, we create boolean variables f_a^{add} and f_a^{del} for each $f \in AddP(a)$ and $f \in DelP(a)$ with corresponding weights $w_a^{add}(f)$ and $w_a^{del}(f)$. Each complete assignment of these variables is a candidate model of E and corresponds to a candidate model of $\mathcal{D}^*$.

SAT constraints: We introduce the notion of *confirmed level* C_f^i for each proposition f needed at level i, which is the latest level j ($j < i$) at which the value of f is confirmed to be either $\mathbf{T}$ (i.e. $f \in Pre(a_j)$ or $f \in Add(a_j)$) or $\mathbf{F}$ (i.e. $f \in Del(a_j)$) by the action a_j. Observing that the truth value of f at level i is affected only by possible effects of actions at levels within $[C_f^i, i-1]$.

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

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

ensuring 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 additive effect. If f is confirmed to be $\mathbf{F}$ at C_f^i, we add an additional constraint to ensure that it is added before i:

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

Possible precondition establishment: When a possible precondition f is realized as a precondition of a_i (i.e. $f_{a_i}^{pre} = \mathbf{T}$), it needs to be established and protected using constraints *C1* and *C2* above. Specifically, if f is confirmed $\mathbf{T}$ at the level C_f^i, we protect this truth value with a variant of *C1*:

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

Otherwise, we establish it with a variant of *C2*:

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

Approximate Assessment The exact computation discussed above has exponential run time in the worst case, and would not be useful when one considers comparing robustness of two given plans, or incorporating robustness assessment into a robust plan generation procedure (see below). We now describe two approximate approaches to estimate plan robustness.

Algorithm 1: Approximate plan robustness.

1 **Input:** The plan $\pi = (a_0, a_1, ..., a_n)$;
2 **Output:** The approximate robustness value of π;
3 **begin**
4 $R_\pi(0) = 1$;
5 **for** $i = 1..n$ **do**
6 **for** $p \in F$ **do**
7 $R_\pi(p, i) \leftarrow$ ApproxPro$(\{p\}, i, \pi)$;
8 $R_\pi(i) \leftarrow$ ApproxAct(i, π);
9 Return $R_\pi(n)$;
10 **end**

Using approximate weighted model-counting algorithms: Given a logical formula representing constraints on domain models with which the plan π succeeds, in this approach an approximate model-counting software, for instance Weighted ApproxCount (Wei and Selman 2005), is invoked to get the approximate number of domain models.

Robustness propagation approach: We are also investigating 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 $R_\pi(i)$ as the robustness value of the action step i, and define the robustness value for any set of propositions $Q \subseteq F$ at level i, $R_\pi(Q, i)$ $(i > 0)$, as the weighted 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, 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).

Generating Robust Plans

Our next contribution will be to investigate various techniques that can be put on top of the FF planner (Hoffmann and Nebel 2001) in order to generate solution plan with high robustness value. Given the current state s_i, reached from a sequence of actions $\pi_i = (a_0, a_1, ...a_{i-1})$, the purpose is to advance the search by choosing one state among k successor states $s_{i_1}, s_{i_2}, ..., s_{i_k}$, taking into account the robustness value (in addition to the reachability measure computed from the relaxed plans $RP(s_{i_j})$ built from s_{i_j}). We briefly describe high-level ideas that we are working on:

Model-counting Approach One simple idea using robustness value in evaluating successor states is first constructing constraints for applicability of actions in the partial plan π_i (as discussed in the previous section) and in the relaxed plan $RP(s_{i_j})$, and then use model counting algorithms to compute the robustness value. This value then can be combined with the the reachability information (i.e. the number of actions in $RP(s_{i_j})$) to evaluate successor states.

Extracting Robust Relaxed Plan This approach aims to modify the construction of the relaxed planning graph and the relaxed plan extraction of FF, from which a relaxed plan more sensitive to incompleteness information of the domain model can be extracted, and therefore can guide the search toward robust solution plans. While details of this approach still need to be worked on, we outline some design issues being considered: (1) the construction of the relaxed planning graph should take into account the robustness information of the partial plan π_i; (2) the propagation of robustness of actions in the relaxed planning graph (which might be different from that of the partial plan); and (3) the termination condition in building the relaxed planning graph needs to change (intuitively, the robustness value of a goal proposition p may still be improved by extending the relaxed planning graph with actions possibly adding p).

Conclusion

This thesis aims to address planning scenarios where the given model of user's preferences or domain dynamics is partially specified, a realistic problem that many planning systems would face, and yet has not been given enough attention in the planning community. We therefore expect that our work will allow planning techniques to enter a broader range of real-world applications, and also hope to extend our methods to other planning formulations (such as temporal planning, contingency planning, etc.)

References

Carlyle, W. M.; Fowler, J. W.; Gel, E. S.; and Kim, B. 2003. Quantitative comparison of approximate solution sets for bicriteria optimization problems. *Decision Sciences* 34(1).

Garland, A., and Lesh, N. 2002. Plan evaluation with incomplete action descriptions. In *Proc. of AAAI-02*.

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs in LPG. *Journal of Artificial Intelligence Research* 20(1):239–290.

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 *Proc. of AAAI-07*.

Mali, A., and Kambhampati, S. 1999. On the utility of plan-space (causal) encodings. In *Proc. of AAAI-99*.

Nguyen, T.; Do, M.; Kambhampati, S.; and Srivastava, B. 2009. Planning with partial preference models. In *Proc. of IJCAI-09*.

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

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