Analyzing Reachability within Plan Space

Romain Trinquart
LAAS-CNRS, 7 Av. du Colonel Roche, Toulouse 31077 Cedex,
romain.trinquart@laas.fr

Abstract

We present a new structure developed to analyze reachability within the plan space. The main goal is to propose a new heuristic estimator to guide the search in partial order planners. It could be used to evaluate establishers of open-conditions as well as resolvents for threats or resource contention. Algorithms are detailed to expand this structure and update it along the search.

1 Introduction

Several recent works have contributed to bring back to light partial-order algorithms. First a number of existing architectures for integrating planning with execution illustrate one reason why partial-order planners remain attractive over state space and CSP- or SAT- based planners : they provide plans which offer a higher degree of execution flexibility. The most striking example is of course NASA's RAX [Johnson *et al.*, 2000]. It has also been shown that the pessimism about the scalability of partial-order planners could be challenged ([Nguyen and Kambhampati, 2001], [Younes and Simmons, 2002]). However, eventhough some important improvements have been achieved, we are still lacking a solid domain-independent heuristic estimator which would guide the search within plan space.

The most impressive results to date are exhibited by planners making use of solid search control information which are derived from the structure proposed in [Blum and Furst, 1995] : the planning graph. The planning graph is an approximation of the part of the state space which can be reached from a given initial situation. It is built in polynomial time. It has been used in different successful fashions. First, it has been proposed as the basis to build plans through its encoding in a CSP or SAT problem ([Koehler *et al.*, 1997], [Kautz and Selman, 1999]). It has also been used as a basis to build informed estimators to evaluate the distance between a node reached during the search and a goal ([Hoffmann and Nebel, 2001]). Last it has been used as a new search space in [Gerevini and Serina, 2002] where the authors expose algorithms to explore a space whose nodes are subgraphs of the planning graph. There has been some attempts to benefit from the strength of the planning graph in a plan space search. But transposing successful approaches in state space is not straightforward and raises some difficulties.

Let us detail the Plan Space exploration process so as to explain this last statement. Nodes of the Plan Space are partial plans which can contain flaws : open conditions (conditions with no supporting actions) or threats (two mutually exclusive propositions which could possibly interfere with each other or cause resource contentions). Moves in this search space consist in flaw repairs. In order to complete the search for a solution, partial plans are refined by the addition of resolvents, either establishers for open conditions (causal link with an action already in the plan or a new action) or constraints (ordering or binding/separation) for threats. Thus choosing how to progress in the search implies ranking very different operators, each one repairing one flaw. As stated in [Weld, 1994], this choice is usually decomposed into two steps : first a flaw is chosen and the neighborhood of the current node is defined by the resolvents of the chosen flaw, then a resolvent is chosen.

Elaborating efficient heuristics to take these decisions is complicated by the partial order on actions that prevent us from determining the context in which actions will actually be applied. Somehow, since it is not obvious to associate a partial plan to a "current state", it is difficult to estimate how far it is from a solution. A good review of the heuristics used to choose which flaw to focus on can be found in [Pollack *et al.*, 1997]. But it turns out that most of these control strategies are problem-dependent. To achieve better performance, VHPOP ([Younes and Simmons, 2002]) periodically switch from one strategy to another along the search. Even if this versatility might pay out, it is not really satisfying since it needs additional control for instance to decide how often the control strategy should be changed.

In [Nguyen and Kambhampati, 2001], the authors focus on the selection of a resolvent for a given open-condition. They propose a novel heuristic estimator relying on the use of a planning graph to rank partial plans. Their planner, RePOP, uses only ground actions and does not handle resource. Thus the only threat's resolvents to be considered are ordering constraints. Instead of branching on either demotion or promotion, RePOP uses a disjunctive representation of orderings during plan refinement. Once all the flaws are solved, the remaining disjunctive constraints are split into the search space. The efficiency of the POP algorithm is reported to be dramatically improved. However this system has some limits. One is the use of disjunctive constraints : extend-

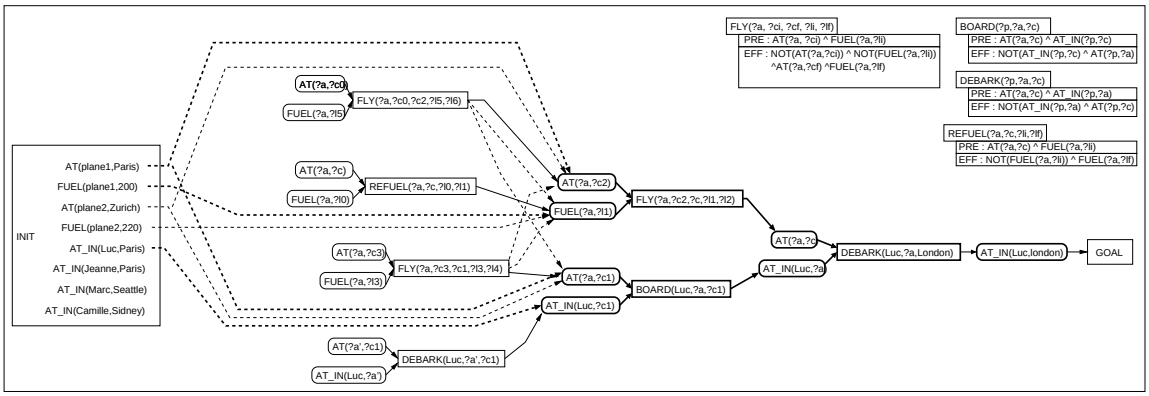

Figure 1: the PS-PG for a problem of the domain Zeno

ing RePOP to handle lifted actions or resources would introduce some threats with more than two alternative resolvents. Keeping such disjunctions would significantly complicate the constraints propagation. Furthermore choosing which open-condition (or flaw) to focus on is still an open issue since RePOP relies on the same classic heuristics as VHPOP.

Our work has been motivated by this lack of a well grounded and uniform search control strategy in plan space. In the following sections of this paper, we first propose a structure that enables us to catch a part of the Plan Space that is relevant to a given problem. We highlight some properties of this structure, mainly that it can reflect the effects of commitment made along plan refinements. Finally we propose to use this structure to obtain control information ranking establishers as well as threat resolvents.

2 The Plan Space Planning Graph : Definitions and Properties

In this section, we first review the basis of partial-order planning algorithms. Then we expose methods to build a disjunctive structure representing a part of the Plan Space. We name it the "Plan Space Planning Graph" (PS-PG).

In this paper we will only consider the context of classical planning problems. The initial state of the world I, the list of goals G and the set of deterministic action schemas Ω are given. Each action schema $a \in \Omega$ has a precondition list and an effect list, denoted respectively $Pre(a)$ and $Eff(a)$. Finding a plan consists in building an ordered set of actions (instantiations of action schema) which will achieve the goals G when executed from I.

Background on POP Algorithm - The Partial Order Planning algorithm searches the solution plan into the partial plan space. A partial plan can be considered as a conjunction of constraints that defines a set of "candidates plans". The search process will reduce this set of candidates until it discards every inconsistent plan, the remaining plans being solutions for the problem.

More formally a partial plan is defined as a 6-tuple $(\mathcal{A}, \mathcal{O}, \mathcal{B}, \mathcal{L}, \mathcal{OC}, \mathcal{UL})$ where $\mathcal{A}$ is a set of potentially ungrounded actions, $\mathcal{O}$ a set of ordering constraints over $\mathcal{A}$, $\mathcal{B}$ a set of constraints over the variables figuring in elements of $\mathcal{A}$ and $\mathcal{L}$ is a set of causal links [1] (i.e. oriented arcs between actions labeled by a proposition which figure commitment for an action to establish one precondition of an other action). The description of a partial plan is completed by two sets of flaws : $\mathcal{OC}$, a set of open conditions, and $\mathcal{UL}$, a set of unsafe links (threats). A partial plan stands for a family of plans. It is considered to be a valid solution if all of these plans are consistent. This implies that the sets $\mathcal{OC}$ and $\mathcal{UL}$ are empty. An open condition designates a non-explained precondition of one of the actions in $\mathcal{A}$. A threat appears when an action a_1 establishes a precondition p of an action a_2 while a third action a_3 might cause the cancellation of this precondition, i.e $not(p)$ is an effect of a_3 and a_3 might occur between a_1 and a_2 without violating any constraint in $\mathcal{O}$.

The algorithm starts its search from a partial plan which contains only two actions A_0 and A_∞. A_0 has only effects: it establishes the initial state. A_∞ has only preconditions: it initializes the open conditions with the desired final state.

Step by step, the POP algorithm is going to try to resolve all the flaws of the partial plan. A step consists in selecting a flaw and choosing a resolvent. An open condition can be explained either by inserting a new action whose effects unify with this open condition or by establishing a new causal link with an action already in the plan. To resolve a threat, it is necessary to over-constrain the actions so as to ensure non-overlapping between the threatening action and the threatened link. This is achieved by posting temporal (promotion/demotion) or atemporal (inequalities) constraints.

If the search reaches a node which contains one flaw with no possible resolvent, it has gone into a dead-end branch : it has to backtrack on one of its previous choices.
The selection of the flaw and the choice of a resolvent involve good heuristics to guide the search. As seen above, the heuristic estimators used in this context should be able to rank various resolvents : intern causal links, causal links with new tasks, ordering constraints and binding or separation constraints. In order to build such an estimator, we propose a new way to compute reachability within Plan Space.

[1] Since we consider ungrounded actions, causal links involve unifying parameters of actions and open-conditions. If $c = a \xrightarrow{p} b$ is a causal link, we will denote $\sigma(c)$ the substitution that maps the variables in b to the variables in a.

The Plan Space Planning Graph - This structure is intended to catch the part of the plan space that might be explored by the POP algorithm. Like a partial plan, it can be considered as a set of actions and constraints defining a family of potential plans. But instead of committing to one resolvent for each flaw, the idea is to keep disjunctions. The relaxation here consists in neglecting threats and focusing on open conditions and their establishers. We define the elements of the structure and how it is expanded. The expansion procedure strongly suggests a succession of open-conditions layers and establishers layers. The resulting structure can be considered as an extension of the Action Graph, described in [Smith and Peot, 1993], to deal with lifted actions.

Definition - The *Plan Space Planning Graph (PS-PG)* is an oriented graph with two kind of nodes : proposition and action nodes. It can be formally expressed as a 7-tuple ($\mathcal{P}$, $\mathcal{A}$, Pre, $\mathcal{EL}$, $\mathcal{IL}$, $\mathcal{O}$,$\mathcal{B}$). $\mathcal{P}$ and $\mathcal{A}$ are respectively proposition and action nodes. Pre are edges from $\mathcal{P}$ to $\mathcal{A}$: if $a \in \mathcal{A}$ then $\mathcal{P}$ contains nodes corresponding to the preconditions of a and Pre contains edges from these nodes to a. $\mathcal{EL}$ and $\mathcal{IL}$ complete the set of edges : they are sets of causal links from actions to propositions ($\mathcal{EL}$ stands for extern link and $\mathcal{IL}$ for intern link). We also denote $\mathcal{L} = \mathcal{IL} \cup \mathcal{EL}$. $\mathcal{O}$ and $\mathcal{B}$ are set of ordering constraints and constraints over variables; these constraints are induced by the causal links in $\mathcal{L}$.

The PS-PG is initialized with the two actions A_0 and A_∞. It is then expanded according to the procedure *"Expansion"*.

Expansion :

1. $OC \leftarrow Pre(A_\infty),\ NewOC \leftarrow \emptyset$
2. while ($OC \neq \emptyset$)
3. for every oc in OC
4. $ExternEst \leftarrow$ supporting_task(oc)
5. for every E in $ExternEst$
6. insert(E)
7. $NewOC \leftarrow NewOC \cup Pre(E)$
8. endfor
9. endfor
10. for every oc in OC
11. $InternEst \leftarrow$ intern_establishers(oc)
12. for every E in $InternEst$
13. insert(E)
14. endfor
15. endfor
16. Add constraints from the new causal links
17. $OC \leftarrow NewOC,\ NewOC \leftarrow \emptyset$
18. endwhile

Notes on the "Expansion" procedure - The function *supporting_task* returns the list of all the possible extern establishers of the given open-condition oc, i.e. new actions which have one effect that can be unified with oc. Inserting an extern establisher consists in the following : binding the variables involved in the action and in the open-condition (new constraints in $\mathcal{B}$), adding the action in $\mathcal{A}$, its preconditions in $\mathcal{P}$ and the corresponding edges in Pre, then a causal link with the open-condition is added in $\mathcal{EL}$.

intern_establisher(oc) returns the list of all the possible intern establishers of oc, i.e. all the actions already in $\mathcal{A}$ whose one of the effects could be unified with oc. Deciding if an element of $\mathcal{A}$ could be an intern establisher of oc involves checking weither this causal link would be consistent with $\mathcal{O}$ and $\mathcal{B}$. Inserting an intern establishers simply entails adding a causal link in $\mathcal{IL}$.

On Line 16, the constraints we are concerned with are orderings and bindings from the new establishments : an action should be ordered after the establishers of its preconditions and the domains of its parameters also depend on its establishers. Since there is no commitment to a particular establisher, these constraints are disjunctive. Given an action $a \in \mathcal{A}$ and one of its precondition $p \in \mathcal{P}$, the following constraints hold :

- $(\bigvee_i (a > b_i), b_i \in Est(p)) \in \mathcal{O}$ where $Est(p) = \{b \in \mathcal{A}, b \rightarrow p \in \mathcal{L}\}$
- $\forall x \in Var(p) \quad Dom(x) \subset \cup_i (Dom(y_i), y_i \in Est(x))$ where $Est(x) = \{y, \exists b \in \mathcal{A} / y = \sigma(b \rightarrow p)(x)\}$

A last remark should be made on the stop condition used on line 2. This is not a realistic stop condition for the simple reason that in most cases a partial plan can always be extended by putting actions into it (loops) which entail more and more open conditions to be established. Instead, the expansion should be stopped when there are reason to believe the PS-PG contains a solution plan. We will propose an actual stop condition in the following sections.

Figure 1 provides an example of a PS-PG developed for a problem in the domain Zeno. There are 4 actions in the domain (Fly, Refuel, Board, Debark). Solid (resp. dashed) lines stand for extern (resp. intern) causal links. Bold lines represent the links that are part of the actual solution of the problem.

Before moving on to using the Plan Space Planning Graph as the basis for a heuristic estimator, let us state one of its important properties.

Property - *All the partial plans that the POP algorithm could explore are contained in the Plan Space Planning Graph.*

More precisely, if the Expansion procedure passes through the main loop N times, then any partial plan that might be refined by the POP algorithm to resolve N open-conditions (and any number of flaws) is contained in the PS-PG.

Intuitively, this property is ensured by the two steps within the main loop of "Expansion". First all new tasks are computed and inserted in the PS-PG. It is only after this that intern causal links are computed. Thus tasks that have just been added to establish an open-condition might also be reused to establish other open-conditions as side-effects. This way we make sure that all combinations of resolvents are considered.

3 Estimating distance within Plan Space

As said above, the PS-PG can be considered as a set of constraints that defines a set of candidate plans. It is possible to evaluate a lower bound of the length of a solution plan through the PS-PG. Adding constraints, like committing to one establishers or enforcing an order between two actions, will reduce the set of candidates and thus potentially change (increase) the lower bound of the length of solution plan. This change can be used as an estimator to rank resolvents in func-

tion of their likeliness to bring the partial plan closer to a solution plan.

In this section we will first present heuristic functions to estimate a lower bound of the length of a solution plan using the PS-PG. We will then present a method to enforce in the PS-PG either a commitment to an establisher or ordering constraints or even binding/separation constraints. We will then dispose of all the necessary tool to build a heuristic search control for a POP algorithm.

Lower bound of the length of a solution plan - These functions compute a lower bound of the length of a solution plan. The length of a partial plan P is measured as the number of refinement that should be applied to the initial partial plan to reach P. It is a strict lower bound since resolvents for threats are not taken into account. The formulas presented below are induced by the interpretation of the PS-PG as an AND/OR structure : all the open-conditions in a partial plan should be established to reach a solution and each open-condition can be established in different ways.

In the following equations, P denotes a partial plan, $OC(P)$ the set of open-conditions in P, p a proposition, $Est(p)$ the set of possible establishers of p which appear in the PS-PG (i.e. $Est(p) = \{a \in \mathcal{A} \,/\, a \rightarrow p \in \mathcal{L}\}$), E a task and finally $OC(E)$ is the set of open-conditions E would introduce in the plan.

$$(1)\ cost(P) = \sum_{p \in OC(P)} cost(p)$$

$$(2)\ cost(p) = \min_{E \in Est(p)} cost(E) \text{ if } Est(p) \neq \emptyset,$$
$$\infty \text{ otherwise}$$

$$(3)\ cost(E) = 1 + \sum_{p \in OC(E)} cost(p)$$

It should be noted that these functions take positive interactions into account. An action is used only once as an extern causal link (see Procedure *Expansion*). If it is used to support other open-conditions as side-effects, it is through an intern causal link. Since intern causal link do not entail any open-condition, the cost of establishing the preconditions will be counted only once, even if it is used to established several propositions.

Last these functions provide a criterion to stop the expansion of the PS-PG : as long as the cost of the initial partial plan is infinite, the POP algorithm will have to refine the partial plan with establishers which are not in the PS-PG. So a strategy consists in expanding the PS-PG so as to reach a finite cost and then consider further expansion if it is required by the search.

Enforcing choices in the PS-PG - The procedure *"Expansion"* proposed in the previous section catches a part of the Plan Space according to the "constraints" defined by the initial situation and the goals. Now we consider methods to modify the PS-PG accordingly to additional constraints : we are interested in determining what would be the consequences of committing to a specific flaw resolvent with respect to the reachable part of the search space.

Committing to an establisher for an open condition will discard all the alternative establishers : it involves discarding the preconditions they entailed and of course it prevents them from being further used in intern causal links. Committing to an ordering or separation constraint to solve a threat will affect some establishers which will no longer be consistent with either $\mathcal{B}$ or $\mathcal{O}$. The procedure *"Enforce"* is presented below to show how these changes are propagated within the PS-PG in case of commitment to an establisher. The procedure for a constraint (threat's resolvent) is basically the same.

Enforce(Establisher E)

1. $a \rightarrow p = E$
2. add constraint $(a > s)$ where $(p \rightarrow s) \in Pre$
3. add binding constraints between a and p
4. $DiscardedEst \leftarrow establishers(p) - a$
5. $DiscardedOC \leftarrow entailed\_oc(DiscardedEst)$
6. Remove DiscardedEst from the PS-PG.
7. $OC \leftarrow Pre(A_\infty)$
8. $while(OC \neq \emptyset)$
9. for every oc in OC
10. $Ests \leftarrow establihers(oc)$
11. if $oc \in DiscardedOC$ then
12. $DiscardedOC \leftarrow entailed\_oc(Ests)$
13. Remove Ests from the PSPG
14. else
15. for every E in $Ests$
16. if $no\_longer\_possible(E)$ then
17. $DiscardedOC \leftarrow DiscardedOC$
18. $\cup\, entailed\_oc(E)$
19. Remove E from the PS-PG
20. else
21. $NewOC \leftarrow NewOC \cup entailed\_oc(E)$
22. endif
23. endfor
24. endif
25. endfor
26. $OC \leftarrow NewOC, NewOC \leftarrow \emptyset$
27. endwhile

Removing an extern causal link $a \rightarrow p$ from the PS-PG consists in removing $a \rightarrow p$ from $\mathcal{EL}$, removing a from $\mathcal{A}$ and for every precondition p of a removing p from $\mathcal{P}$ and removing $p \rightarrow a$ from Pre. Removing an intern causal link simply consists in removing it from $\mathcal{IL}$.

On line 16, checking if a causal link is no longer possible consists in checking if the involved action has not been discarded from the PS-PG and if it is still consistent with the current constraints (ordering and binding/separation).

On line 8, this stop condition is sufficient : contrary to what is done in *"Expansion"*, here we are not computing new establishers but retrieving what is already in the PS-PG. Thus, since the expansion of the PS-PG has finished, we are sure to stop.

Search control - We are now ready to explain how to use the PS-PG to guide the search process in a POP algorithm. The basic idea is that at each step of the search resolvents will be ranked according to the lower bound of the length of a solution computed via the PS-PG once they have been enforced. The PS-PG will be incrementally updated along the search by actually enforcing each resolvent which is chosen to be added to the current partial plan. The procedure POP

below gives a naive sketch of algorithm to illustrate such a search control.

POP(I,G)
1. $P \leftarrow$ initial partial plan $(\{A_0, A_\infty\}, \{A_0 < A_\infty\}, \emptyset, Pre(A_\infty), \emptyset)$
2. $Flaws \leftarrow Pre(A_\infty)$
3. Expand the PS-PG
4. while ($Flaws \neq \emptyset$)
5. $Resolvents \leftarrow compute\_resolvent(Flaws)$
6. if ($Resolvents = \emptyset$) then Backtrack.
7. for every R in $Resolvents$
8. Evaluate the cost of R in the PS-PG
9. /* i.e. temporary enforce R and compute $cost$ */
10. endfor
11. choose R with minimal cost
12. add R in P and update $Flaws$
13. if P is inconsistent then Backtrack.
14. enforce R in the PS-PG
15. endwhile

This simple setting enables us to highlight one of the strength of this heuristic estimator : it evaluates in a uniform way all kinds of resolvents. Thus it could be used in a much more expressive context, for instance in systems handling time and numeric resources such as IxTeT ([Ghallab and Laruelle, 1994]) or HSTS ([Muscettola, 1994]). What would then be required are modules to analyze flaws and elaborate their resolvents. For instance such methods as those exposed in [Laborie, 2001] could be used to build an efficient Resource Contention Detection module that would elaborate sets of constraints to resolve these new threats. These constraints could then be evaluated through the PS-PG.

Beyond the control schema exposed above, we could elaborate a more complex algorithm. First we could mix it with classical POP settings, such as a two-level choice : first select a flaw and then a resolvent. But the PS-PG itself could be used more actively. For instance, it could be used to compute the resolvents for open-conditions at each step of the search. Furthermore instead of inserting establishers one at a time, deterministic branches of the PS-PG (i.e part of the graph with no alternatives) could be inserted altogether.

4 Conclusion

We have presented a structure to encode the part of the Plan Space which might be explored by a POP algorithm for a given problem, namely the Plan Space Planning Graph. We propose to use it as a basis from which to compute heuristic information and to build a search control strategy.

We are currently working on the effective implementation and experimentation of the search control proposed in section 3. We have implemented the PS-PG within the IxTeT system. Although very preliminar, the results we obtain indicate that the heuristic turns out to be much more efficient than fixed strategy such as LIFO, Z-LIFO or LCFR. However expanding the PS-PG remains expensive and hardly bears comparison with the systems which took part in the third International Planning Competition. We are continuing to work on the implementation so as to check whether the extended temporal context might be responsible for these results and find out how to make this approach computationally effective.

References

[Blum and Furst, 1995] Avrim Blum and Merrick L. Furst. Fast planning through planning graph analysis. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.

[Gerevini and Serina, 2002] Alfonso Gerevini and Ivan Serina. Lpg: a planner based on local search for planning graphs. In *Proceedings of the International Conference on AI Planning Systems*, 2002.

[Ghallab and Laruelle, 1994] M. Ghallab and H. Laruelle. Representation and Control in Ixtet, a Temporal Planner. In *Proceedings of the International Conference on AI Planning Systems*, 1994.

[Hoffmann and Nebel, 2001] J. Hoffmann and B. Nebel. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14, 2001.

[Johnson *et al.*, 2000] A. Johnson, P. Morris, N. Muscettola, and K. Rajan. Planning in interplanetary : Theory and practice. In *Proceedings of the International Conference on AI Planning Systems*, 2000.

[Kautz and Selman, 1999] Henry Kautz and Bart Selman. Unifying sat-based and graph-based planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1999.

[Koehler *et al.*, 1997] J. Koehler, B. Nebel, J. Hoffmann, and Y. Dimopoulos. Extending planning graphs to an adl subset. In *Proceedings of the European Conference on Planning (ECP)*, 1997.

[Laborie, 2001] Phillipe Laborie. Algorithms for propagating resource constraints in ai planning and scheduling : Existing approaches and new results. In *Proceedings of the European Conference on Planning (ECP)*, 2001.

[Muscettola, 1994] N. Muscettola. Hsts: Integrated planning and scheduling. In *Fox, M., and Zweben, M., eds, Intelligent Scheduling, Morgan Kaufman*, 1994.

[Nguyen and Kambhampati, 2001] XuanLong Nguyen and Subbarao Kambhampati. Reviving partial order planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2001.

[Pollack *et al.*, 1997] Martha Pollack, David Joslin, and M. Paolucci. Flaw selection strategies for partial-order planning. *Journal of Artificial Intelligence Research*, 6, 1997.

[Smith and Peot, 1993] David E. Smith and Mark Peot. Postponing threats in partial-order planning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1993.

[Weld, 1994] Daniel S. Weld. An introduction to least commitment planning. *Artificial Intelligence*, 15, 1994.

[Younes and Simmons, 2002] Hakan Younes and Reid Simmons. On the role of ground actions in refinement planning. In *Proceedings of the International Conference on AI Planning Systems*, 2002.