Pruning Methods For Optimal Delete-free Planning Using Domination-Free Reachability

Avitan Gefen and **Ronen I. Brafman**
Department of Computer Science
Ben-Gurion University of The Negev, Israel

Abstract

Delete-free planning (DFPlan) underlies many popular relaxation (h^+) based heuristics used in state-of-the-art planners, and a number of recent planning domains are naturally delete-free. This has led to increased interest in efficient methods for cost-optimal DFPlan. To aid in the solution of DFPlan problems, we introduce a new analysis technique, called domination-free reachability (DFR). DFR is an improved version of the well known notion of reachability in graphs that filters out nodes that are not useful for optimal planning. We explain how to compute DFR, and present three new pruning techniques that use it. Combined with a recent decomposition technique these pruning methods lead to effective pruning in delete-free planning.

Introduction

Heuristic search is currently the preferred method for solving satisficing and optimal planning problems. Among alternative methods for generating heuristic functions, relaxation-based heuristic functions are extremely popular, effective, and influential (Hoffmann and Nebel 2001; Helmert and Domshlak 2009). These methods estimate minimal distance-to-goal of a state in the delete-free problem generated from a given instance by removing all delete-effects. Computing this value, known as h^+, is NP-hard (Bylander 1994). Effective approximations exist (Hoffmann and Nebel 2001; Helmert and Domshlak 2009), yet efforts continue to improve these techniques in order to provide even better heuristic estimates (Pommerening and Helmert 2012; Haslum, Slaney, and Thiébaux 2012; Gefen and Brafman 2012), as well as to handle naturally delete-free domains more effectively (Gefen and Brafman 2011; Porco, Machado, and Bonet 2011; Gallo et al. 1993).

In this paper we investigate improved pruning techniques for delete-free planning (DFPlan). Our pruning techniques rely on a new construction, *domination-free reachability* (DFR) that seeks to improve the standard notion of reachability for optimal planning. Reachability and reachability analysis, whether forwards or backwards, underlies some fundamental techniques in planning. DFR attempts to provide a more refined notion of reachability, one that filters out "useless" nodes in the context of optimal planning. A node is useless in optimal planning if it is not part of some optimal plan from the initial state to the goal. Obviously, computing whether a node is part of an optimal plan is a difficult problem, and DFR which is a practical, polynomial time technique cannot compute this. What DFR provides is an approximation of this concept, whose practical utility is shown in our empirical evaluation.

The DFR technique operates on a faithful graphical depiction of a DFPlan problem, called the relaxed causal graph (Keyder, Richter, and Helmert 2010; Gefen and Brafman 2012). The nodes of this graph correspond to facts and actions. Given special source node (that corresponds to the initial state) and a target node (goal state) DFR seeks to find all nodes that are part of at least one minimal plan from the initial state to the goal. A plan π for G is *minimal* if no other plan for G is a strict subset of the set of actions in π. That is, we can not remove any action from π without loosing its legality or its ability to achieve G. Finding the set of all nodes which are part of a minimal plan is NP-hard, and the DFR is an over-approximation of it that is often much smaller than the set obtained using standard reachability.

We focus on the notion of minimal plans because of the results of (Gefen and Brafman 2012) (GB). GB show that, given a set of landmarks suitably ordered, an optimal plan can be generated by generating minimal plans between these landmarks. This implies that one can decompose the planning problem into multiple simpler search problems in each of which we seek to achieve a landmark, and where one *needs only consider minimal plans.* The DFR helps us make the search for each such minimal plan more efficient because it allows us to prune nodes that are guaranteed to not belong to any minimal plan.

In the first part of this paper, we define the notion of DFR, show how to compute it, and demonstrate its soundness with respect to minimal plans. In the second part of the paper, we show how to use the DFR to prune A* search for an optimal plan. We discuss three methods. The first, the simplest, the most obvious, and the most effective simply prunes actions that are not part of the (backwards) DFR of the goal. The second technique is a simpler variant of the notion of disjoint-path commitment introduced by GB. Intuitively, if a_1 and a_2 are two actions that achieve the goal then a minimal plan will include only one of them. If we can identify that our current plan is a prefix of a minimal plan targeting

a_1, due to minimality, we can commit to a_1, and prune actions that are only part of a minimal plan targeting a_2. The DFR algorithm, which provides a sound estimate of "being in a minimal plan" allows us to operationalize this intuition. Finally, in the last pruning method we describe, we use the DFR algorithm to provide an upper bound on the cost of a minimal plan because we know that the true minimal plan is a subset of the set computed by DFR. This bound can be used for standard bound-based pruning. We conclude the paper with an empirical evaluation of these pruning methods and a short discussion.

Delete-Free Planning Tasks

A *delete-free STRIPS planning task*, or *DF task* for short, is a 4-tuple (P, A, I, G). P is a finite set of *propositions*. A state s is represented by the set of propositions that are *true* in it. $I \subseteq P$ is the *initial state*. $G \subseteq P$ is the set of propositions that must be true at any *goal state*. A is the set of *actions*, where $a \in A$ has the form: $a = \langle pre(a), add(a) \rangle$ denoting its preconditions and add effects. An action a is applicable in a state $s \subseteq P$ iff $pre(a) \subseteq s$. Applying a in s transforms the system to the state $s \cup add(a)$. We use $a(s)$ to denote the resulting state. When a is not applicable in s then $a(s)$ is undefined. We do *not* consider actions with conditional effects. Therefore, there is always an optimal plan with only one instance of each action, as a second instance of an action has no effect on the state.

A *solution* to a DF task is a plan $\pi = (a_1, \ldots, a_k)$ such that $G \subseteq a_k(\cdots(a_1(I))\cdots)$. That is, it is a sequence of actions that transforms the initial state into a state satisfying the goal conditions.

Landmarks play an important role in the work of GB: they are used to decompose the problem. A *fact landmark* for a state s is a proposition that holds at some point in every plan from state s to the goal (Hoffmann, Porteous, and Sebastia 2004). An *action landmark* for a state s is an action that must be part of any plan from s to the goal. A *disjunctive action landmark* for a state s is a set of actions, at least one of which must be part of every plan from state s to the goal (Helmert and Domshlak 2009).

A plan π for G is *minimal* if no other plan for G contains a strict subset of the actions in π. Clearly, any optimal plan must be minimal, unless zero-cost actions exist, in which case it must have a sub-plan which is minimal. Therefore, when seeking an optimal plan for G, we can prune any plan that is not minimal without sacrificing optimality.

The following Lemma underlies the decomposition technique of GB.

Lemma 1 (Gefen & Brafman, 2012)**.** *Let L be a set of fact landmarks for a DFPlan problem $\Pi = (P, A, I, G)$, such that $G \subseteq L$. Then, if there is a solution to Π, there exists an ordering $l_1, \ldots, l_k$ of L and a minimal plan π for Π such that $\pi = \pi_1, \ldots, \pi_k$, where π_i is a minimal plan for $(P, A, \pi_{i-1}(\cdots(\pi_1(I))\cdots), l_i)$.*

Building on existing efficient algorithms for landmark detection (Keyder, Richter, and Helmert 2010), GB provide an efficient procedure for generating a landmark ordering as needed in Lemma 1. Thus, instead of planning for G, one can incrementally plan for each of the landmarks, keeping around minimal plans only. Although this method may prune some optimal plans, it is guaranteed to leave some optimal plans intact.

For the first part of the paper, we will focus on the task of finding one of the component minimal plans π_i. This could be a plan to reach l_1 from I, or a plan to reach l_k from $\pi_{k-1}(\cdots(\pi_1(I))\cdots)$. The reader can simply think of it as reaching G' from some initial state I'.

The Relaxed Causal And/Or graph

We can capture the structure of delete-free problems using a directed And/Or graph known as the *relaxed causal graph* (RCG) (Keyder, Richter, and Helmert 2010; Gefen and Brafman 2012). An And/Or graph is a directed graph with two types of nodes: *And* nodes and *Or* nodes. The RCG $\mathcal{G}$ associates an *And* node with each action and an *Or* node with each fact. There is an edge from (the node for) fact p to action a if $p \in pre(a)$, and an edge from a to p if $p \in add(a)$. In addition, there are two special *Or* nodes and two special *And* nodes: s the start node (Or), is attached to a special *And* node i via a special initial state edge, (s, i), and t (Or) is attached to a special *And* node g via a special goal edge, (g, t). There is an edge from s to every action with no preconditions. There is an edge from s to i and an edge from i to every initial state fact. There is an edge from every goal fact to g, and an edge from g to t. See Figure 1 for an example. To keep the generality of planning terms for the RCG we define the following: $pre(i) = s$, $pre(a) = s$ for every action a without a precondition, $add(i) = \{\text{initial state facts}\}$, $pre(g) = \{\text{goal facts}\}$, $add(g) = t$.

A subgraph $J = \langle V^J, E^J \rangle$ of $\mathcal{G}$ is a DF plan (aka, *justification subgraph*) iff the following holds:

1. It contains s and t.
2. For every *And* node $a \in V^J$, it contains all incoming edges and the *Or* nodes from which they originate.
3. For every *Or* node $p \in V^J \setminus \{s\}$, it contains at least one incoming edge and the node it originates from.
4. J is acyclic.

These conditions ensure that there is an action achieving each proposition (as well as t) and that each action has all its preconditions satisfied. The acyclicity ensures the plan is well founded.

Domination-Free Reachability

Forwards and backwards reachability analysis is one of the central methods used in planning. In particular, structured reachability analysis underlies popular techniques such as planning graphs (Blum and Furst 1997).

In this section, we define the notion of domination-free reachability (DFR) in an RCG. The DFR set for a node v, $dfr(v)$. Ideally, we would have liked $dfr(v)$ to contain only fact nodes n that are part of a minimal plan from s to v. Unfortunately, this would be too hard to compute:

Lemma 2. *Finding the set of all nodes which are part of a minimal plan is NP-hard.*

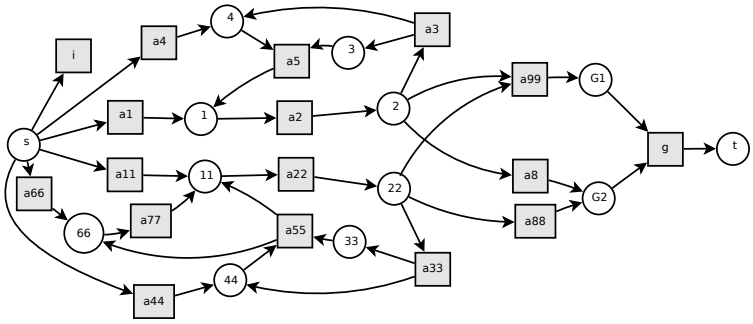

Figure 1: An RCG. Facts in white, actions in grey, G1 and G2 are the goal and hence fact landmarks, g is a special goal action. $Dom(2) = \{s, 1, 2\}$, $Dom(3) = \{s, 1, 2, 3\}$, $Dom(22) = \{s, 11, 22\}$, $Dom(33) = \{s, 11, 22, 33\}$. $dfr(s) = \{s\}$, $dfr(1) = \{s, 1\}$, $dfr(2) = \{s, 1, 2\}$, $dfr(3) = \{s, 1, 2, 3\}$, $dfr(4) = \{s, 1, 2, 3, 4\}$, $dfr(11) = \{s, 11, 44, 66\}$, $dfr(22) = \{s, 11, 22, 44, 66\}$, $dfr(33), dfr(44), dfr(66) = \{s, 11, 22, 33, 44, 66\}$

Proof. The hardness of this problem comes from the fact that a minimal plan is a generalization of the concept of a simple path in control flow graphs. (imagine a planning problem that can be represented using only a directed graph). The question of whether there exist a simple path from s to t with x as an obligatory node is equivalent to the 2-disjoint paths problem (Perl and Shiloach 1978) which is known to be NP-complete. □

Instead, $dfr(v)$ conservatively approximates (= is a superset of) the set of fact nodes that can be part of a minimal plan from the start node s to v. That is, all nodes in $dfr(v)$ lie on some path from s to v, but not all fact nodes on such paths necessarily belong to $dfr(v)$. That is, some nodes that would be obtained by standard reachability analysis for v are not part of $dfr(v)$. Hence, domination-free reachability is a *filtered* form of reachability that can prune certain nodes that are not part of a minimal plan.

To the best of our knowledge, it is the first effective variant of reachability geared towards optimal planning. We will define the notion of DFR, an algorithm to compute it, and demonstrate it through an example. We will show that it is an over-approximation, i.e., that it can contain nodes that are not part of a minimal plan, but that it can be much smaller than the standard set of reachable nodes.

Domination

Our procedure is based on the notion of *domination* used in *control flow* graphs: "Vertex v dominates vertex w in flow graph $(G; s)$ if $v \neq w$ and every path from s (start node) to w contains v" (Tarjan 1974). We adapt the idea of domination to And/Or graphs as follows: a fact v dominates a fact w in the RCG $\mathcal{G}$ if every DF plan from s to w contains v. Intuitively, one can understand this as saying that every plan from the current state that achieves w achieves v, as well. Slightly departing from the definition in control-flow graphs, our definition also implies that v dominates itself.

Landmarks are often used in planning to refer to goal dominators, but also, sometimes, to dominators in general. We will use the original "dominators" term because it comes with less baggage, it is more appropriate linguistically for our setting, and it is related to the graph-theoretic ideas that motivate us. We use $Dom(x)$ (called the *Dom set* of x) to denote all fact nodes that dominate fact x in the RCG, including x itself. This set could also be referred to as landmarks for x. For a set X, we define $Dom(X) = \bigcup_{x \in X} Dom(x)$. The process used to find landmarks in (Keyder, Richter, and Helmert 2010) computes the Dom sets naturally.

We use the notion of domination to refine the notion of reachability. To do so, we must formalize the notion of a path in the RCG. We borrow the following paths definitions from directed-hypergraphs (Gallo et al. 1993): A path $p = (v^0, a^1, v^1, a^2, v^2, \ldots, a^k, v^k)$ in the RCG is a sequence of facts and actions, where v^0 is the origin node, v^k is a target node and for every $1 \leq i \leq k$, $v^{i-1} \in pre(a^i)$ and $v^i \in add(a^i)$. A *simple path* is a path in which no action appears twice. A simple path will be *elementary* if no fact appears twice.

To better illustrate the advantage of DFR we define $reach(v)$, which captures the standard notion of reachability. $reach(v)$ is the set of action nodes that are part of some RCG path from the start node s to v. This set can be obtained by a backwards traversal of the RCG (treated as a regular directed graph) that starts at v and collects nodes along this traversal. Alternatively, one can do a forward traversal starting at s. These procedures can collect action nodes that are not part of a minimal plan to v.

For example, in Figure 1 $reach(1) = reach(2) = reach(3) = reach(4) = \{a_1, a_2, a_3, a_4, a_5\}$. In contrast to these sets, we can see that the only minimal plan from s to 2 will contain action nodes a_1, a_2 and fact nodes $s, 1, 2$. Indeed, node 3 is dominated by node 2.

If we can identify this fact, we can ignore node 3 when exploring the space of possible minimal plans from s to 2. This will be done (to some extent) by the DFR algorithm below.

DFR Algoritm

Algorithm 1, described above takes the RCG graph for the initial state ($\mathcal{G}(I)$) at its input, and its output are the DFR sets for each fact node. It iteratively propagates information forward, and can be understood as reachability analysis with filtration.

Algorithm 1 Domination-Free-Reachability

```
1: DFR(G(I))  {G(I) is initial state RCG}
2: for each v ∈ V do dfr(v) ← {v}
3: Q ← {s}, R ← {s}  {s is the start node}
4: while Q ≠ ∅ do
5:    Select and remove v ∈ Q
6:    for all a ∈ consumers(v) do
7:       if pre(a) ⊆ R then
8:          for all ef ∈ add(a) do
9:             R ← R ∪ {ef}
10:            if ef ∉ Dom(pre(a)) then
11:               if add(a) ⊈ dfr(ef) do Q ← Q ∪ {ef}
12:               dfr(ef) ← dfr(ef) ∪ add(a)
13:               for all pc ∈ pre(a) do
14:                  for all r ∈ dfr(pc) do
15:                     if ef ∉ Dom(r) and r ∉ dfr(ef) then
16:                        dfr(ef) ← dfr(ef) ∪ {r}
17:                        Q ← Q ∪ {ef}
18:                     end if
19:                  end for
20:               end for
21:            end if
22:         end for
23:      end if
24:   end for
25: end while
```

Let $dfrA(v) = \{a |\ pre(a) \cup add(a) \subseteq dfr(v)\}$ be the set of actions which their related fact nodes are in the $dfr(v)$. Then, for each $v \in V$, $dfrA(v) \subseteq reach(v)$. For example, in Figure 1 $dfr(1) = \{s, 1\}$ and $dfr(2) = \{s, 1, 2\}$, showing the superiority of the DFR sets over regular reachability. Unfortunately, a formal characterization of $dfr(v)$ is non-trivial – we discuss it in a longer version of this paper which is in preparation. Roughly speaking, $dfr(v)$ holds all facts which are part of a minimal plan to v. These are nodes that can reach v (or can be reached while achieving v) but which satisfy the additional constraint that they are not dominated by v. There are two exceptions: (1) v itself is dominated by v, and yet it is part of $dfr(v)$. (2) Nodes that appear together with v in the add effect of some action.

Technically, the algorithm works as follows: For every *Or* node v, we initialize $dfr(v)$ to v, since any node reaches itself as long as there is a plan that achieves it. Two other sets are Q – our propagation set, and R – the reachability set. Both are initialized with the initial state node s. Now, we start the iterative process (l. 4): To propagate information forward, we remove a fact node v from Q and go over all actions that can consume v (v is one of their precondition) (l. 6). For each such action a, if all of a's preconditions were already reached (l. 7), we can try to use a to propagate information to its effect nodes. Therefore, we iterate over a's effect nodes ef (l. 8) as follows: First, we update the reachability set R with ef – the node we just reached (l. 9). Now, our first filtration is based on the fact that information should be propagated to ef only if ef does not dominate a (i.e., any of its preconditions). Otherwise, any plan that achieves ef using a (ef is already true before applying a) is not minimal (l. 10). If ef does not dominate a, we would like to add $add(a)$ to $dfr(ef)$, since they can be achieved simultaneously with ef (l. 11,12). Finally, we can go over all DFR sets of the preconditions pc of a. For each node $r \in dfr(pc)$, as long as ef does not dominate r, we can propagate r to $dfr(ef)$ (l. 13–16). When we propagate new information we update Q so the new information can be propagated further (l. 11 & 17).

Let us examine an example using Figure 1. The RCG in Figure 1 holds two similar sub-graphs: one that is composed of nodes 1, 2, 3, 4 and the other 11, 22, 33, 44, 66. The fact landmarks are nodes 1, 2, 11, 22, G1, G2. A backward procedure that collect actions starting at fact node 1 will collect actions a_1, a_2, a_3, a_4, a_5. Two of these actions are applicable in the initial state (a_1, a_4). On the other hand $dfr(1) = \{s, 1\}$, and therefore only action a_1 is in $dfrA(1)$ (since node 1 is a fact landmark and a_1 is the only action that can achieve it in a minimal plan, we can conclude it is an action landmark). This outcome results from the fact that node 1 is a dominator of 3, and therefore a_5 cannot propagate information into node 1.

Unfortunately, the DFR algorithm is not perfect as can be seen using node 11. Action a_{55} cannot propagate information to node 11, but it will propagate information to node 66. Action a_{77} will propagate nodes 44,66 to node 11 and so $dfr(11) = \{s, 11, 44, 66\}$. With node 44 in $dfr(11)$ also actions a_{44}, a_{55} will be part of $dfr(11)$ although there is no minimal plan to 11 with those actions.

Correctness

The following Lemma ensures the correctness of DFR sets as a super-set of minimal plans for some fact node.

Lemma 3. *Let $dfr(G')$ be the DFR set computed using Algorithm 1 for DFPlan problem $\Pi = (P, A, I', G')$. If for some action a, $pre(a) \cup add(a) \not\subseteq dfr(G')$ then there is no minimal plan for G' that contains action a.*

Proof. Paths in the RCG are the channels for information propagation in Reachability/DFR algorithms. Every plan (justification graph) can be seen as a union (of actions) of, not necessarily disjoint, simple paths. The number of actions in each simple path that is contained in some plan π must be $\leq$ the length of the plan. Therefore, the length of simple paths can serve as a lower bound to the length of a minimal plan.

Notice, that if π is a minimal plan, then for any simple path p "extracted" from π that preserves the order of π, we have that for any actions $a_i, a_j \in p$ where a_j appear after a_i, a_j cannot dominate a_i. This follows from the definition of a minimal plan. We now prove by induction on the length of simple paths that the DFR sets include all fact nodes which are part of a minimal plan.

(i) Our induction assumption is that when we covered all paths of length i, if there is a minimal plan with length $\leq i$ to x that includes a, then $pre(a) \cup add(a) \subseteq dfr(x)$.

(ii) The base case, $i = 0$, is immediate since $dfr(s)$ includes s and only s.

(iii) Assuming the induction hypothesis holds for length i we show it holds at length $i + 1$. Let us look at some fact node x. Let us assume there is a minimal plan π_x of length $i + 1$

to x that ends with a_x. A minimal plan to x must end with an action where $x \in add(a_x)$, moreover it can't achieve x before. If we remove a_x from π_x we may get a plan which is not minimal but is composed of paths with length $\leq i$.
We shall now look at two scenarios, one in which some action e is not an achiever of x (x is not an effect of e), and the second one where it is:
(1) Let us look at some action e which is not an achiever of x (ef in line 8). If $pre(e), add(e) \in dfr(pc)$ where pc is a precondition of a_x (an achiever of x) and x is not a dominator of $pre(a_x)$ (line 10) then we would try to propagate $pre(e), add(e)$ ($r \in dfr(pc)$) to x (line 13–16). If x dominate $y \in pre(e)$ (line 15) then each plan that uses e must reach x before applying e and therefore it is not minimal. x can dominate $y \in add(e)$ without dominate the preconditions of e if it appears with y in all add effects of actions that achieve y (Lemma 4), but this contradicts the fact that e is not an achiever of x.
(2) If $e = a_x$ is an achiever of x and x does not dominate $pre(e)$ (ef in line 10), then in the DFR algorithm $add(e)$ will be added to $dfr(x)$ (line 12), and since each precondition of e will propagate itself to x, since they are not dominated by x (line 13–16) we get that $pre(e), add(e) \in dfr(x)$.

At the end of the run of algorithm DFR we will propagate information for all possible minimal plans (and possibly more). □

Lemma 4. *Let v, x be some facts in a DFPlan problem. Let $A_x \subseteq A$ be a set of actions that achieve fact x (i.e. if $a_x \in A_x$ then $x \in add(a_x)$). If v dominates x, then: (i) It also dominates some precondition of each $a_x \in A_x$, or (ii) In every minimal plan to x where v does not dominate some precondition of $a_x \in A_x$ (the action that achieves x in that plan) it appears with x in the add effect i.e. $x, v \in a_x$.*

Proof. If v dominates $x \in add(a_x)$, every plan that achieves x must also achieve v. If v must be achieved before x, i.e. before some action a_x that achieves x is being applied then v must dominate some precondition of a_x. If there are plans where v is not achieved before x, then it must be achieved simultaneously with x, i.e. $x, v \in add(a_x)$. □

Pruning Using DFR

What follows are three pruning methods that strongly utilize the DFR algorithm. These methods can be applied to any DFPlan problem, but they are even more effective when combined with the decomposition method of GB. The reason for this is that GB's decomposition yields multiple planning problems, each with a shorter solution, and our pruning methods are more effective the closer the goal is to the initial state.

Basic DFR Pruning

The simplest and empirically most powerful pruning method is based on Lemma 3. Given a planning problem with initial state I' and goal state G', prune any action a such that $pre(a) \cup add(a) \not\subseteq dfr(G')$. Lemma 3 guarantees that such actions are not part of a minimal plan to G'.

Path-Commitment using DFR

One of the two pruning methods introduced by GB is called disjoint-path commitment. There, the last action a_{last} applied during A* search towards some goal l, is used to prune actions that are not part of a minimal plan containing a_{last}. To use this method, one must be able to recognize that certain actions belong to different, disjoint minimal plans that lead to l. GB use the following sound approximation to apply this idea in practice: (i) At preprocess time, define a set of labels – one for each action achieving l. Using a backwards traversal from l, propagate these labels backwards. Intuitively, one now identifies different path to the goal with their last action. Let $lbl(a)$ be the set of labels reached to action a. Intuitively, these labels mark the different path to l to which a belongs. (ii) Then, during search, as long as a_{last} did not achieve a fact landmark, one can safely prune any action a where $lbl(a) \cap lbl(a_{last}) = \emptyset$ – that is, the labels of that action are disjoint from the labels of the last action applied.

We will show how to use the DFR sets to obtain an improved version of GB's disjoint path commitment. It replaces GB backwards label propagation – essentially backwards reachability – with the use of DFR sets. As these sets provide a better form of reachability for minimal plans, the resulting method is more powerful. The basic idea is simple: First, associate every action in $dfr(G')$ with one or more preconditions of the actions that achieve the current goal, or landmark. We do so by "reverse" computing DFR sets of these actions.

Next, imagine we have only two actions a_p, a_q with preconditions p, q respectively, which achieve G'. If the action a_{last} that was last applied belongs to the DFR set of precondition p, but not of q, then any minimal plan containing a_{last} must achieve the goal via a_p. It would be redundant (non-minimal) for such a plan to contain a_q as well. Moreover, we can ignore any other action that is not part of a minimal plan containing a_p after applying a_{last}.

Note that the above intuitions are correct provided that the last action applied, a_{last} does not achieve some landmark l. Our discussion below is under this assumption.

We start formulating the ideas above using the following two observations: Let $\pi_i = (a_1, \ldots, a_k)$ be a minimal sub-path to l, where a_k achieves l. (1) We know that $a_1, \ldots, a_k \in dfrA(l)$ because all actions of a minimal plan must be in $dfrA(l)$. (2) We know that $a_1, \ldots, a_{k-1} \in dfrA(x_1) \vee \cdots \vee dfrA(x_t), x_i \in pre(a_k)$, because the first $k-1$ actions must be in the dfrA sets of one of a_k's preconditions.

Notice, that if we knew the identity of a_k, we could prune all actions that are not in the DFR sets of one of a_k's preconditions. We usually cannot know who is a_k – it must belong to the set of achievers of l – but using a_{last} we can sometimes find a smaller set than $ach(l)$. Since we assumed that a_{last} does not achieve a fact landmark, it is one of the actions $a_1, \ldots, a_{k-1}$, as in the observations above. Let us assume the index of a_{last} is a_j ($j \leq k-1$). We can use a_{last} to identify which actions in $ach(l)$ could be in a minimal plan with a_{last}. That is, we can find which actions can act as a_k. Since we know what the possible a_k's are, we can

prune actions that cannot be a_{j+1} in such a plan.

To use these ideas we do the following: (i) Let $dfrb(v) = \{x|v \in dfr(x)\}$. That is, it is the set of facts x such that v can appear in a minimal plan that reaches them. (ii) Let $dfrb(a) = \bigcap_{v \in pre(a) \cup eff(a)} dfrb(v)$. $dfrb(a)$ is the set of facts such that a can be part of a minimal plan that reaches them. (iii) Let $PRE(l) = \{f|f \in pre(a) \wedge a \in ach(l)\}$, i.e., all preconditions of actions that achieve l. (iv) We can now define $cmt(a_{last}, l) = dfrb(a_{last}) \cap PRE(l)$. This is a set of fact commitments for l given a_{last}. At least one of these facts must be achieved in any minimal plan for l that includes a_{last}. That is, it is similar to a disjunctive landmark focused on minimal plans for l that include a_{last}.

Let us examine Figure 1 again, and assume our next landmark to achieve is G1 and that $a_{last} = a_1.dfrb(a_1) = \{s, 1, 2, 3, 4, G1, G2, t\}$, $PRE(G1) = \{2, 22\}$. Therefore $cmt(a_1, G1) = \{2\}$. We can notice three things: (1) We eventually must be able to apply action a_{99} (2) We must achieve node 2 (which is in $cmt(a_1, G1)$) (3) We also need to achieve node 22 (which is not in $cmt(a_1, G1)$). Since, we must achieve node 2, we can focus our search on this node until it is achieved and then turn our focus to node 22. We now formulate this observation

During A* search we divide the commitment set $cmt(a_{last}, l)$ into two sets. The first set $cmt_1(a_{last}, l)$ composed of the facts in the commitment set still unachieved. $cmt_2(a_{last}, l)$ is composed of achieved facts (facts in the state). The second set must be strengthened as follows: $cmt_2^*(a_{last}, l) = \{x|x \in pre(a) \wedge pre(a) \cap cmt_2 \neq \emptyset \wedge a \in ach(l)\}$. That is, we add to this set all preconditions of actions that uses facts from cmt_2 and achieve l. The reason this strengthening is necessary is because that cmt_2 could hold only part of the preconditions of some action in $ach(l)$ (as in the example above). If that precondition is already in the current state (like node 2) we will still need to achieve the rest of the preconditions (node 22). Now, during search, if a_{last} did not achieve a fact landmark, and action $a \notin ach(l)$, we can safely prune a, if there is no fact y in cmt_1 or cmt_2^* where $a \in dfrA(y)$. For example, if now $a_{last} = a_2$, then cmt_1 will be empty, $cmt_2 = \{2\}$ and $cmt_2^* = \{2, 22\}$, forcing us to turn our focus to node 22.

Let us now look at another example from Figure 1. If we will remove from the RCG in Figure 1 action a_{99} and fact node G1, we will get an RCG with only one fact landmark G2 (besides s). Now (using the mentioned RCG), let us assume the last action taken was a_1 which did not achieve a fact landmark. $dfrb(a_1) = \{s, 1, 2, 3, 4, G2, t\}$, $PRE(G2) = \{2, 22\}$. Therefore $cmt(a_1, G2) = \{2\}$. So, we can prune any action that is not part of $dfrA(2)$. Notice, that if a_8 would have another precondition besides node 2, we could miss it based on $dfrb(a_1)$ alone. In that case, We could safely prune until node 2 is in state, then we would have to strengthen cmt_2 to "see" the other precondition.

Sub-goal Bounds

Our third and last pruning technique uses the DFR differently. Let us assume we have some upper bound u on the cost of the entire plan to G. We'd like to get an upper bound on the cost of achieving G', our current landmark. We do so by first estimating the state $s(G')$ reached when G' is achieved. Then using an admissible heuristic to obtain a lower bound d on the cost of achieving the final goal G from $s(G')$. Now, $u - d$ is an upper bound on the cost of achieving G'. We use the $dfr(G')$ as our estimate of the state $s(G')$. This is justified because we know that all facts achieved on route to G' using some minimal plan are included in $dfr(G')$. This gives us a "quick and dirty" method of generating an upper bound on the cost of achieving G' which requires only a single computation of the heuristic function (very cheap) and a single computation of $dfr(G')$, which is computed anyway during preprocessing.

An upper bound u on plan cost provides a simple (well known) method for pruning during A^* search. Any state s such that $f(s) = g(s) + h(s) > u$ can be pruned (assuming an admissible h value). Our main observation here is that this idea can be used at each stage implemented in the GB framework, since their decomposition approach reduced the problem of searching for a single plan to one of searching for many (simpler) plans (see Lemma 1).

Unlike the first two pruning methods discussed so far, which would work for any I', G', the bounds generation method takes a slightly more global perspective, and will generate the upper bounds for all the relevant subproblems obtained in GB's decomposition at preprocessing time.

First, we get an upper bound on the cost of the entire plan. To get an upper bound on plan cost, one can use h^{add}, h^{max} (Bonet and Geffner 2001) to guide the selection for a satisficing plan, much like in FF (Hoffmann and Nebel 2001). A stronger technique will be to use algorithms described in (Keyder and Geffner 2009). The cost of this plan is an upper bound u. Next, recall that GB's decomposition method generates a sequence $L = (l_1, l_2, \ldots, l_k)$ of ordered landmarks (see Lemma 1). Using this sequence, we create a sequence of partial assignments that an optimal plan must reach. These partial assignments are supersets for minimal plans, specifically, the plan to l_1 must reside in $dfr(l_1)$.

Since in DFPlan propositions achieved remain true forever, as long as we keep the landmark ordering, a minimal plan to $\{l_1, l_2\}$ must lead to a state in which the achieved propositions are a subset of $dfr(l_1) \cup dfr(l_2)$. More generally, the superset of any state achieving l_j is $dfr(l_1) \cup \cdots \cup dfr(l_j)$.

Using an admissible heuristic function h, we now get the following respective bounds on the cost of reaching each of the landmarks: Let $B = (u - h(dfr(l_1)), u - h(dfr(l_1) \cup dfr(l_2)), \ldots, u - h(dfr(l_1) \cup \cdots \cup dfr(l_k)))$, where $h(dfr(v))$ is an admissible heuristic of state $s = \{x|x \in dfr(v)\}$ to the goal. This computation needs to be carried out once only, before search commences.

During search, for each current state s, each applicable action a, and next landmark to achieve l_i, we can ask: is $g(s) + cost(a) \leq B(i)$? If the answer is *no*, we can prune a. If all actions applicable at s are pruned, s is a dead-end for optimal planning.

Empirical Results

We implemented the ideas described in the previous sections in the Fast Downward framework (Helmert 2006). The DFR

algorithm was implemented to run once at preprocess time. The Q set in the algorithm was implemented as a queue. Pruning itself is done while in search, with respect to the next fact landmark to achieve. We implemented two variants: DFR1 uses the first pruning method only (basic DFR pruning). DFR2 uses all three pruning methods. We evaluated their performance in the context of heuristic search (using A^*) using LM-Cut and compare to the results in the GB paper in Table 1,2. Domains used are delete-free versions of IPC problems. The time limit is 0.5/5/30 minutes per problem. We compare LM-Cut alone to GB + LM-Cut, DFR1 + LM-Cut and DFR2 + LM-Cut. The overall performance (coverage, time scores, expansion scores) shows that pruning in general has an advantage over LM-Cut alone in almost all of the domains. There are some domains where the GB method is better than the basic pruning (DFR1), like logistics98 and satellite. For the satellite domain this can be the result of the path-commitment based pruning, since, DFR2 is better than the GB method in this domain. For logistics98, the GB method is better than DFR2 so this is probably due to minimization of disjunctive action landmark that is done in the GB method for which we have no parallel technique (although it is theoretically possible). We show DFR2 with all three pruning methods because our experiments showed that the contribution of sub-goal bounds is minor. Still, we believe it to be an interesting general technique that could be enhanced using stronger heuristics.

We also tried our pruning procedure on the seed-set problems with zero-cost actions. Zero-cost actions are challenging to existing solvers which cannot solve any instance of this problem (Gefen and Brafman 2011). Pruning, in this case, is not sufficient to overcome this method. However, when we augment pruning with a simple procedure that applies all applicable zero-cost actions, then pruning+blind-search+zero-cost-procedure solves all the seed-set problems.

Domain	LM-Cut			GB method			DFR1 + LM-Cut			DFR2 + LM-Cut			
Seconds	30	300	1800	30	300	1800	30	300	1800	10	30	300	1800
airport(50)	26	33	38	40	48	49	29	42	49	23	28	43	48
blocks(35)	35	35	35	35	35	35	35	35	35	35	35	35	35
depot (22)	4	6	7	10	10	12	9	10	10	9	9	10	10
driverlog (20)	13	14	14	13	14	15	14	15	15	13	14	15	15
elevators-opt08-strips (30)	5	7	7	6	7	9	5	7	8	4	5	6	8
freecell (80)	1	4	6	0	1	1	1	4	6	1	1	4	6
grid (5)	1	1	2	1	1	2	1	2	2	1	1	2	2
gripper (20)	20	20	20	20	20	20	20	20	20	20	20	20	20
logistics00 (28)	23	23	23	28	28	28	28	28	28	28	28	28	28
logistics98 (35)	7	8	8	10	17	17	12	15	16	11	13	16	16
miconic (150)	150	150	150	128	150	150	150	150	150	150	150	150	150
mprime (35)	19	24	27	17	21	24	19	24	27	17	20	25	28
mystery (30)	23	25	26	20	25	26	22	26	26	17	23	26	26
openstacks-opt08-strips (30)	5	6	7	5	7	8	4	5	7	30	30	30	30
openstacks-strips (30)	5	5	5	5	5	5	5	5	5	5	5	5	5
parcprinter-08-strips (30)	23	23	23	29	29	30	30	30	30	30	30	30	30
pathways--noneng (30)	5	5	5	5	7	8	5	5	8	5	5	6	8
pegsol-08-strips (30)	24	27	27	24	26	29	25	27	27	18	25	27	27
pipesworld-notankage (50)	12	15	17	10	13	17	12	15	17	10	12	16	17
pipesworld-tankage (50)	7	10	10	6	7	9	7	10	10	6	7	10	10
psr-small (50)	50	50	50	50	50	50	50	50	50	50	50	50	50
rovers (40)	8	12	12	17	20	21	18	20	21	20	21	25	28
satellite (36)	6	6	7	6	8	10	6	7	7	6	7	9	11
scanalyzer-08-strips 30)	13	14	15	8	10	14	10	14	15	9	10	14	15
sokoban-08-strips 30)	19	21	25	22	24	26	20	22	26	17	20	22	26
tpp (30)	9	12	13	14	15	16	17	21	21	19	19	21	21
transport-opt08-strips (30)	9	10	12	9	12	12	9	10	12	9	10	10	10
trucks-strips (30)	6	7	9	24	30	30	30	30	30	30	30	30	30
woodworking-opt08-strips 30)	17	18	19	27	27	28	27	27	29	27	27	27	29
zenotravel (20)	11	13	13	13	13	13	13	13	13	13	13	13	13
Total (1116)	556	604	632	602	680	714	633	689	720	633	668	725	752

Table 1: Coverage per domain. time limit per problem: 0.5 (30s), 5 (300s), 30 (1800s) minutes (sec.). We add a DFR2 column of 10 seconds to see where it is tied with 30 minutes of LM-cut alone.

Summary and Related Work

Using the graph-theoretic notion of domination, we were able to construct a new simple domination-free reachability method for delete-free planning and use it within the decomposition framework of GB to obtain better performance than the more complicated pruning techniques of (Gefen and Brafman 2012). In addition, we leveraged the decomposition to introduce a bounds-based pruning method. Our method was shown to be more effective in solving delete-free planning problems. The increased coverage is not large – we solve 16 more problems than GB – yet in the context of optimal planning, this is considered non-trivial progress. Perhaps more importantly, our work is based on a new construction that improves upon one of the more fundamental tools in the analysis of planning problems – that of reachability – and we are not aware of any similar such construct in the literature.

The DFR algorithm shares some similarity with the *first achiever analysis* (Haslum, Slaney, and Thiébaux 2012; Pommerening and Helmert 2012) which removes all operators that are not *first achievers* of any variable (Richter, Helmert, and Westphal 2008). Such actions can not be part a minimal plan. In fact, Line 10 in the DFR algorithm prevents the propagation of information through actions which are not first achievers, treating them as if they were not in the RCG. The DFR algorithm then continues to detect actions that cannot be part of a minimal plan but still would not be removed by the first achiever analysis. For example, in Figure 2 the DFR set of nodes $1, G_1$ does not contain actions a_2, a_3 which will not be part of a minimal plan even though they are first achievers.

A similar notion to path-commitment in regular planning was recently introduced by (Karpas and Domshlak 2012). In this paper causal links are used to infer constraints that must be satisfied by an optimal plan having some known prefix. The constraints are used to enhance a heuristic evaluation. Therefore, the notion of optimal plan plays a similar role as the minimal plan in the path-commitment method. It could be the case that using the notion of optimal plan is too restrictive, and using minimal plan instead would simplify this work.

To our knowledge, the work of GB and this paper are the only work in the literature that focuses on pruning in delete-free problems using A* search. There are, however, some other recent works on delete-free planning: (Pommeren-

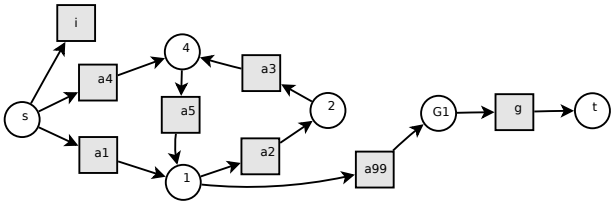

Figure 2: An RCG where first achiever analysis would not remove actions (but i).

planner	Time score	Expansion score
LM-cut	46.58	45.75
GB method + LM-cut	53.05	57.07
DFR1 + LM-cut	54.82	55.59
DFR2 + LM-cut	58.36	58.89

Table 2: time limit per problem: 30 minutes. Scores are average of domain score values for each planner. Scores based on (Richter and Helmert 2009). Logarithmically scaled scores between 0 ($\geq 300s$ and ≥ 1000000 expansions resp.) and 100 ($\leq 1s$ and ≤ 100 expansions resp.).

ing and Helmert 2012) share some similarities regarding upper-bounds, although they don't use A* search. (Haslum, Slaney, and Thiébaux 2012) try to solve delete-free problems by generating minimal disjunctive action landmarks and then applying minimal-hitting set procedure to find an optimal plan. There has been also quite a few recent papers dealing with action pruning: Stratified Planning (SP) (Chen, Xu, and Yao 2009), Expansion core (EC) (Chen and Yao 2009), Bounded intention Planning (BIP) (Wolfe and Russell 2011), Symmetries (Coles and Coles 2010; Fox and Long 1999; Pochter, Zohar, and Rosenschein 2011). Of these, the only work we are aware of that is landmark based is the SAC algorithm (Xu et al. 2011) and its use of disjunctive action landmarks is closer in spirit to GB.

The DFR sets in this paper are generated at preprocessing time giving them an advantage over the GB method which is repeatedly applied at run-time. This can be seen in the fact that even though GB use an extra minimization step to generate minimal disjunctive action landmark, the basic DFR method (DFR1) has, in general, a better average time score, while DFR2 has also a better average expansion scores. These observations supports our belief that the DFR set is a strong theoretical concept, and we believe that it can be used by other methods – any method that can benefit from the notion of minimal plans rather than (any) plan could potentially benefit from the use of DFR sets.

Acknowledgements: We are grateful for the extreme detailed and useful comments made by the anonymous reviewers and also to Malte Helmert and Florian Pommerening for their help in experiments. The authors were partly supported by the Paul Ivanier Center for Robotics Research and Production Management, and the Lynn and William Frankel Center for Computer Science.

References

Bacchus, F.; Domshlak, C.; Edelkamp, S.; and Helmert, M., eds. 2011. *Proceedings of the 21st International Conference on Automated Planning and Scheduling, ICAPS 2011, Freiburg, Germany June 11-16, 2011*. AAAI.

Blum, A., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artif. Intell.* 90(1-2):281–300.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artif. Intell.* 129(1-2):5–33.

Bylander, T. 1994. The computational complexity of propositional strips planning. *Artif. Intell.* 69(1-2):165–204.

Chen, Y., and Yao, G. 2009. Completeness and optimality preserving reduction for planning. In *IJCAI*, 1659–1664.

Chen, Y.; Xu, Y.; and Yao, G. 2009. Stratified planning. In *IJCAI*, 1665–1670.

Coelho, H.; Studer, R.; and Wooldridge, M., eds. 2010. *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings*, volume 215 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Coles, A. J., and Coles, A. 2010. Completeness-preserving pruning for optimal planning. In Coelho et al. (2010), 965–966.

Fox, M., and Long, D. 1999. The detection and exploitation of symmetry in planning problems. In Dean, T., ed., *IJCAI*, 956–961. Morgan Kaufmann.

Gallo, G.; Longo, G.; Pallottino, S.; and Nguyen, S. 1993. Directed hypergraphs and applications. *Discrete Applied Mathematics* 42(2-3):177–201.

Gefen, A., and Brafman, R. I. 2011. The minimal seed set problem. In Bacchus et al. (2011).

Gefen, A., and Brafman, R. I. 2012. Pruning methods for optimal delete-free planning. In McCluskey et al. (2012).

Gerevini, A.; Howe, A. E.; Cesta, A.; and Refanidis, I., eds. 2009. *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*. AAAI.

Haslum, P.; Slaney, J. K.; and Thiébaux, S. 2012. Minimal landmarks for optimal delete-free planning. In McCluskey et al. (2012).

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In Gerevini et al. (2009).

Helmert, M. 2006. The fast downward planning system. *J. Artif. Intell. Res. (JAIR)* 26:191–246.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: fast plan generation through heuristic search. *J. Artif. Int. Res.* 14(1):253–302.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *J. Artif. Intell. Res. (JAIR)* 22:215–278.

Karpas, E., and Domshlak, C. 2012. Optimal search with inadmissible heuristics. In McCluskey et al. (2012).

Keyder, E., and Geffner, H. 2009. Trees of shortest paths vs. steiner trees: Understanding and improving delete relaxation heuristics. In Boutilier, C., ed., *IJCAI*, 1734–1739.

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and complete landmarks for and/or graphs. In Coelho et al. (2010), 335–340.

McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds. 2012. *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*. AAAI.

Perl, Y., and Shiloach, Y. 1978. Finding two disjoint paths between two pairs of vertices in a graph. *J. ACM* 25(1):1–9.

Pochter, N.; Zohar, A.; and Rosenschein, J. S. 2011. Exploiting problem symmetries in state-based planners. In Burgard, W., and Roth, D., eds., *AAAI*. AAAI Press.

Pommerening, F., and Helmert, M. 2012. Optimal planning for delete-free tasks with incremental lm-cut. In McCluskey et al. (2012).

Porco, A.; Machado, A.; and Bonet, B. 2011. Automatic polytime reductions of np problems into a fragment of strips. In Bacchus et al. (2011).

Richter, S., and Helmert, M. 2009. Preferred operators and deferred evaluation in satisficing planning. In Gerevini et al. (2009).

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In *AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence*, 975–982. AAAI Press.

Tarjan, R. E. 1974. Testing flow graph reducibility. *Journal of Computer and System Sciences* 9(3):355–365.

Wolfe, J., and Russell, S. J. 2011. Bounded intention planning. In Walsh, T., ed., *IJCAI*, 2039–2045. IJCAI/AAAI.

Xu, Y.; Chen, Y.; Lu, Q.; and Huang, R. 2011. Theory and algorithms for partial order based reduction in planning. *CoRR* abs/1106.5427.