Simplifying Planning Tasks with Fact-Level Relevance Analysis

Cameron Allen[1], Anita de Mello Koch[2], Harsha Kokel[3], George Konidaris[2], Michael Katz[3]

[1]UC Berkeley [2]Brown University [3]IBM

camallen@berkeley.edu, {anita_de_mello_koch,gdk}@brown.edu, {harsha.kokel,michaelkatz1}@ibm.com

Abstract

Planning problems described in formal languages often contain information that is irrelevant to reaching the specified goal, either by design (to account for multiple possible goals) or as a result of partially automated construction. Consequently, modern planners prune (some) extraneous information by performing safe relevance analysis of state variables prior to search, improving search performance. We show that much more aggressive pruning is possible by reasoning about relevance not at the level of variables, but rather variable *values*, i.e. *facts*. Our approach iteratively identifies relevant facts whenever they appear in the task goal or in the precondition of a relevant action, and identifies relevant actions whenever they achieve a relevant fact (rather than when they simply modify a relevant variable). This already reduces task size relative to variable-level analysis, but it also offers several opportunities for further algorithmic improvements. We formally prove that fact-level relevance analysis, and each subsequent algorithmic improvement we introduce, preserves all shortest optimal plans, without introducing new ones. We show empirically that our approach significantly reduces task size and consequently planning time across a wide range of planning tasks.

Code — https://github.com/camall3n/fact-pruning/

Introduction

A planning agent must either learn or be supplied with a world model rich enough to express any task the agent might encounter during its operational lifetime. Consider a planning agent within the game of Minecraft. Mastering the environment requires hundreds of diverse skills for accomplishing behaviors like navigation, resource gathering, crafting various items, fighting, managing hunger, and decorating.

As the agent's world model grows in complexity to account for new tasks, so too does the cost of planning with it. If the agent is to efficiently accomplish a particular task, like obtaining an axe, it cannot afford to consider all the ways it might bake a cake along the way (see Figure 2, left). Such actions are out of scope for that task and ought to be pruned. However, for another task, where the goal is explicitly to bake a cake, the food-related actions might become essential whereas the axe-related actions become irrelevant. Deciding which parts of the world model to consider for a given task requires either manual human effort or automated analysis to remove extraneous information prior to search.

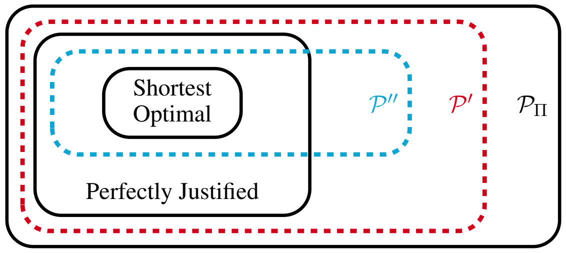


Figure 1: The space of all plans $\mathcal{P}_\Pi$ for a given task. Our algorithms produce subsets of plans $\mathcal{P}'$ and $\mathcal{P}''$ that preserve perfectly justified and shortest optimal plans, respectively, without introducing new ones.

Existing approaches that automatically prune irrelevant information from the agent's world model try to strike a balance between pruning overhead and the search effort it saves (Nebel, Dimopoulos, and Koehler 1997; Boutilier, Brafman, and Geib 1998; Brafman 2001; Hoffmann and Nebel 2001b). Most modern planners base their analysis on the Fast Downward planning system (Helmert 2006), which performs causal graph analysis of state variables to safely and efficiently approximate which variables are goal-relevant. However, this process is overly conservative in the information it prunes.

We build on these approaches, and show that additional relevance analysis can significantly reduce search effort without adding substantial overhead. We argue that reasoning at the level of facts, as opposed to variables, offers several advantages (Fig. 2, right). First, fewer actions will be marked as causally connected to the goal, since action preconditions and effects have fewer facts in common than they do variables. Second, certain causal connections can be ignored if a precondition fact appears in the initial state and is unthreatened by any relevant action. Third, by merging actions with the same effects, we can form a reachability-preserving abstract transition system such that additional pruning is possible and fact relevance can be safely ported back to the original task. Finally, these types of pruning can unlock further pruning benefits from relaxed (forward) reachability analysis, which may in turn enable additional relevance pruning, and so on.

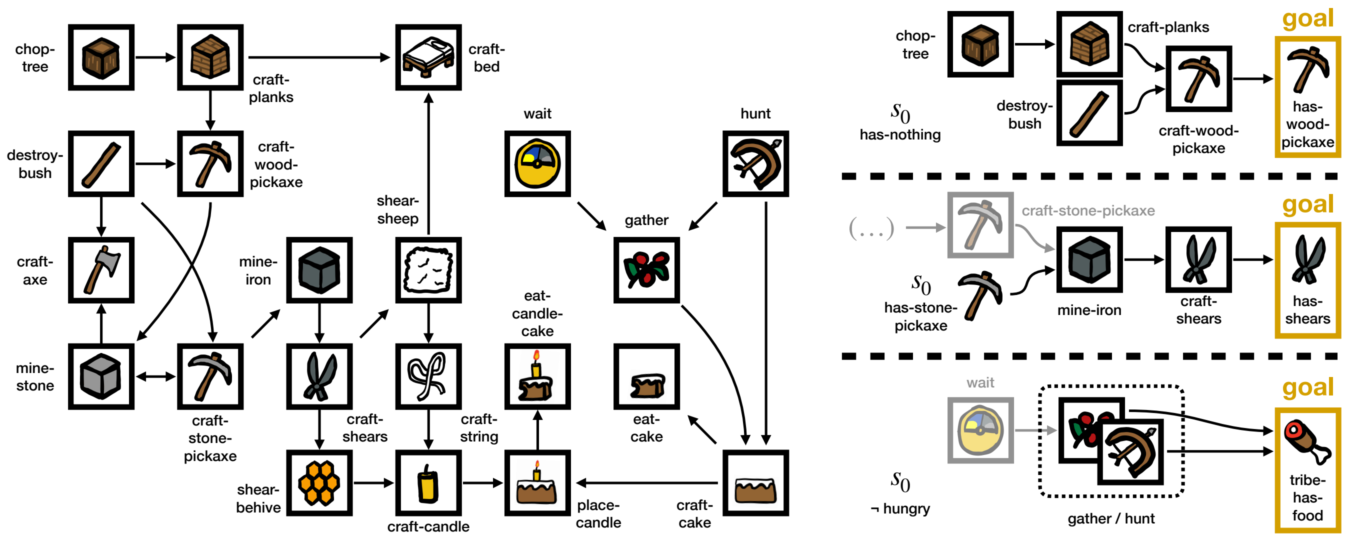


Figure 2: (Left) Action relevance graph for the Minecraft domain. An arrow $a_1 \to a_2$ indicates that $\mathit{eff}(a_1) \cap \mathit{pre}(a_2) \neq \emptyset$. (Right) Three examples of task pruning: (top) relevance analysis regresses backward from the goal wherever actions are linked by at least one fact; causal links to s_0 (middle) and action merging (bottom) allow pruning many more actions.

Our contributions are as follows. We formalize our approach for tasks expressed using a finite domain representation (FDR) (Helmert 2009) and prove that each of these types of pruning is safe: all preserve shortest optimal plans (Katz, Röger, and Helmert 2022), and most variants also preserve perfectly justified plans (Fink and Yang 1992, 1993) (see Fig. 1). We provide an example task in which each type of pruning removes successively more information. We then perform extensive evaluations on benchmark planning tasks taken from the International Planning Competition (McDermott 2000; Long and Fox 2003; Coles et al. 2012; Vallati et al. 2015; Torralba and Pommerening 2018) and on a new set of Minecraft-inspired planning tasks that we designed to showcase our method in a realistic setting. We measure the relative contribution of each of the types of pruning we introduce, and show that, both individually and overall, they lead to significant reductions in problem size and search time, even after accounting for time spent pruning prior to search.

Background

In the SAS$^+$ formalism (Bäckström and Nebel 1995), a *planning task* $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, G, cost \rangle$ consists of a finite set of finite-domain *state variables* $\mathcal{V}$, a finite set of *actions* $\mathcal{A}$, an *initial state* s_0, a *goal* G, and a *cost* function. Each variable $v \in \mathcal{V}$ is associated with a finite domain $\mathcal{D}(v)$ of values. An assignment of $d \in \mathcal{D}(v)$ to $v \in \mathcal{V}$ results in a fact, denoted by $\langle v,d \rangle$. If $f = \langle v,d \rangle$ is a fact, then $\mathcal{V}(f) = v$ gives the variable for that fact. A *partial state* p maps a subset of variables $\mathcal{V}(p) \subseteq \mathcal{V}$ to values in their domains. For a variable $v \in \mathcal{V}$ and a partial state p, the value of v in p is denoted by $p[v]$ if $v \in \mathcal{V}(p)$ and is *undefined* otherwise. A full state (or simply *state*) s maps all variables $\mathcal{V}(s) = \mathcal{V}$ to values in their domains. The set of all states is denoted by $\mathcal{S}$. A partial state p can be constructed from a partial or full state q by considering a restriction $q[\mathcal{V}_p]$ to only the variables in $\mathcal{V}_p \subseteq \mathcal{V}(q)$. State s is *consistent* with a partial state p if they agree on all variables in $\mathcal{V}(p)$, denoted by $p \subseteq s$. Each action a in $\mathcal{A}$ is a pair $\langle \mathit{pre}(a), \mathit{eff}(a) \rangle$, where $\mathit{pre}(a)$ and $\mathit{eff}(a)$ are partial states called *precondition* and *effect*, respectively. Furthermore, a has an associated non-negative cost denoted by $cost(a) \in \mathbb{R}^{0+}$. An action a is applicable in state s if $\mathit{pre}(a) \subseteq s$. Applying a in s results in a state denoted by $s[\![a]\!]$, where $s[\![a]\!][v] = \mathit{eff}(a)[v]$ for all $v \in \mathcal{V}(\mathit{eff}(a))$, and $s[\![a]\!][v] = s[v]$ for all other variables. An action sequence $\pi = \langle a_0, \ldots, a_{n-1} \rangle$ is applicable in state s if there are states $s_0, \ldots, s_n$ such that $s = s_0$, a_i is applicable in s_i and $s_i[\![a_i]\!] = s_{i+1}$ for $0 \leq i \leq n-1$. We denote s_n by $s[\![\pi]\!]$. An action sequence with $G \subseteq s_0[\![\pi]\!]$ is called a *plan*. The cost of a plan π, denoted by $cost(\pi)$ is the sum of the costs of the actions in the plan. The set of all plans is denoted by $\mathcal{P}_\Pi$, an *optimal* plan is a plan in $\mathcal{P}_\Pi$ with the minimum cost, and a *shortest optimal* plan (Katz, Röger, and Helmert 2022) is a plan in $\mathcal{P}_\Pi$ with the minimum number of actions among all optimal plans.

An action in a plan is *justified* if it establishes a fact that (1) was not already established and (2) remains true until it appears in either the goal or the precondition of a subsequent action in the plan.

Definition 1 (Justified Action) *Let $\pi = \langle a_0, a_1, \ldots, a_{n-1} \rangle$ be a plan for a planning task Π, $s_0, \ldots, s_n$ be its corresponding state sequence, and a_i, $0 \leq i \leq n-1$, be some action in π. The action a_i is* justified *for plan π if there exists some variable $v \in \mathcal{V}(\mathit{eff}(a_i))$ such that:*

(i) $s_i[v] \neq \mathit{eff}(a_i)[v]$, and either
(ii) $G[v] = \mathit{eff}(a_i)[v]$ and $s_j[v] = \mathit{eff}(a_i)[v]$ for all $j > i$ or
(iii) $\mathit{eff}(a_i)[v] = \mathit{pre}(a_j)[v]$ for some $j > i$ and $s_k[v] = \mathit{eff}(a_i)[v]$ for all $i < k \leq j$.

An action in a plan is redundant, or *unjustified*, if it can be removed from the plan without invalidating the plan. A plan where each action is justified is called a justified plan. A plan without a redundant subsequence is called perfectly justified plan (Fink and Yang 1992; Nebel, Dimopoulos, and Koehler 1997; Salerno, Fuentetaja, and Seipp 2023). We define justified plans and plan reduction as follows.

Definition 2 (Justified Plan) *A plan π, for the planning task Π, is a* justified plan *if and only if all its actions are justified.*

Definition 3 (Plan Reduction) *Let π be a plan for a planning task Π. A subsequence ρ of π is a* plan reduction *of π if and only if ρ is also a plan for Π.*

Definition 4 (Perfectly Justified Plan) *A plan π, for the planning task Π, is a* perfectly justified plan *if and only if there is no plan reduction of π.*

All shortest optimal plans are perfectly justified plans, and all perfectly justified plans are justified plans. The reverse direction does not necessarily hold for either relationship.

Preserving Action Applicability

Our approach starts by identifying a set of actions that the pruned task ought to retain, then it removes all *other* actions, along with any non-essential facts. This section introduces Algorithm 1 (PRUNE), which describes, for an arbitrary subset of actions, how to retain a sufficient set of facts for computing action applicability. Later sections will discuss how to compute the set of relevant actions.

Theorem 1 *Given a task $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, G, cost \rangle$, a set of actions $\bar{\mathcal{A}} \subseteq \mathcal{A}$, and an action sequence $\pi = \langle a_0, \dots, a_{n-1} \rangle$ that only uses actions in $\bar{\mathcal{A}}$, let $\Pi' =$ PRUNE$(\Pi, \bar{\mathcal{A}})$. Let π' be the corresponding action sequence in Π' with the same names and preconditions. Then π' is a plan for Π' if and only if π is a plan for Π.*

Proof:

Let F be computed as in lines 1-4 of Algorithm 1. For each action $a \in \bar{\mathcal{A}}$, denote its corresponding action in $\mathcal{A}'$ as a', and observe that $\mathit{eff}(a') = F \cap \mathit{eff}(a)$ and that $\mathit{pre}(a') = \mathit{pre}(a)$.

The initial state is preserved fully, so $s_0 = s'_0$ and $\mathit{pre}(a_0) \subseteq s_0 \iff \mathit{pre}(a'_0) \subseteq s'_0$, hence Π' preserves the applicability of the first action.

Now suppose that actions 0 through $i \leq k$ are applicable in both tasks: $a_{0:i}$ and $a'_{0:i}$ are both applicable in s_0. Let $s'_{i+1} = s_0[\![a'_{0:i}]\!]$ and $s_{i+1} = s_0[\![a_{0:i}]\!]$.

Case 1: If π is a plan, we have $\mathit{pre}(a_{i+1}) \subseteq s_{i+1} = s_i[\![a_i]\!]$, which also holds if we project states and preconditions onto the facts in F: $\mathit{pre}(a_{i+1}) \cap F \subseteq s_{i+1} \cap F = s_i[\![a_i]\!] \cap F$. And since $\mathit{pre}(a_{i+1}) \subseteq F$, it follows that $\mathit{pre}(a'_{i+1}) = \mathit{pre}(a_{i+1}) \cap F \subseteq s_i[\![a_i]\!] \cap F = (s_i \cap F)[\![a'_i]\!] = s'_i[\![a'_i]\!] = s'_{i+1}$.

Case 2: If π' is a plan, we have $\mathit{pre}(a'_{i+1}) \subseteq s'_{i+1} = s'_i[\![a'_i]\!]$. Therefore, $\mathit{pre}(a_{i+1}) = \mathit{pre}(a_{i+1}) \cap F = \mathit{pre}(a'_{i+1}) \cap F \subseteq s'_{i+1} \cap F = s_{i+1} \cap F \subseteq s_{i+1}$.

In both cases, the next actions, a_{i+1} and a'_{i+1}, are applicable in states s_{i+1} and s'_{i+1}, as are the action sequences $a_{0:i+1}$ and $a'_{0:i+1}$ in s_0, respectively. By induction on k, both plans are applicable and lead to $s_n = s_0[\![\pi]\!]$ and $s'_n = s_0[\![\pi']\!]$, respectively.

Case 1: If π is a plan, $G \subseteq s_0[\![\pi]\!]$, and we have $G = G \cap F \subseteq s_0[\![\pi]\!] \cap F = s_0[\![\pi']\!]$, so π' is a plan for Π'.

Case 2: If π' is a plan, we have $G = G \cap F \subseteq s'_n \cap F = s_n \cap F \subseteq s_n$, so π is a plan for Π. □

Algorithm 1: PRUNE

```
INPUT:   $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, G, cost \rangle$ // Planning task
         $\bar{\mathcal{A}}$ // Set of actions to preserve
OUTPUT:  $\Pi'$ // Pruned planning task
1:  $F = G \cup \bigcup_{a \in \bar{\mathcal{A}}} \mathit{pre}(a)$ // Precondition and goal facts
2:  $\bar{\mathcal{V}} = \mathcal{V}(F)$ // Relevant variables
3:  $F = F \cup \bigcup_{a \in \bar{\mathcal{A}}} \mathit{eff}(a)[\bar{\mathcal{V}}]$ // Add relevant effect facts
4:  $F = F \cup s_0$ // Add initial state facts
5:  $\mathcal{A}' = \emptyset$ // Initialize new action set
6:  for $a_i \in \bar{\mathcal{A}}$ do
7:      $a'_i = \langle \mathit{pre}(a_i), \mathit{eff}(a_i) \cap F \rangle$
8:      $cost'(a'_i) = cost(a_i)$
9:      $\mathcal{A}' = \mathcal{A}' \cup a'_i$ // Add to new action set
10: end for
11: return $\Pi' = \langle \mathcal{V}, \mathcal{A}', s_0, G, cost' \rangle$
```

Corollary 1 *Each plan in the pruned task Π' corresponds to a plan in the original task Π.*

Proof:

This follows directly from Theorem 1, since $\bar{\mathcal{A}} \subseteq \mathcal{A}$. □

After pruning, two optimizations are possible. First, we can remove variables that have only one value. Second, if we are only concerned with preserving optimal or perfectly justified plans, we can also prune actions with empty effects. We omit these optimizations to make the proofs simpler, but neither changes the underlying transition system.

Theorem 1 guarantees that (1) any plans we can express with the identified subset of actions will remain plans (and have the same cost) in the pruned task and (2) any plans we identify in the pruned task are also plans in the original task. The former ensures solution quality; the latter, solution validity. What remains is to choose which actions to keep.

From Variable-Level to Fact-Level Analysis

We start from a method that is equivalent to the removal of *irrelevant variables* (Helmert 2006).

In RELEVANCE-V (Algorithm 2), we prune the task by performing (relaxed) backward reachability analysis from the goal. We start by marking the goal variables as relevant. We iteratively find actions that affect a relevant variable and mark their precondition variables as relevant. We stop when there is no change to the set of marked variables. The resulting set of actions can then be passed to PRUNE (Algorithm 1) to simplify the task (see previous section).

Algorithm 2: RELEVANCE-V (Variable Relevance)

```
INPUT:   $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, G, cost \rangle$ // Planning task
OUTPUT:  $\bar{\mathcal{A}}$ // Relevant actions

1: $\mathcal{V}^{(0)} = \mathcal{V}(G)$ // Initialize relevant variables with goal
2: $\mathcal{A}^{(0)} = \emptyset$ // Initialize relevant actions with empty set
3: for $i = 1, 2, \dots$ do
4:     $\mathcal{A}^{(i)} = \mathcal{A}^{(i-1)} \cup \{a \in \mathcal{A} \mid \mathcal{V}(\mathit{eff}(a)) \cap \mathcal{V}^{(i-1)} \neq \emptyset\}$
5:     $\mathcal{V}^{(i)} = \mathcal{V}^{(i-1)} \cup \bigcup_{a \in \mathcal{A}^{(i)}} \mathcal{V}(\mathit{pre}(a))$
6:     if $\mathcal{V}^{(i)} = \mathcal{V}^{(i-1)}$ and $\mathcal{A}^{(i)} = \mathcal{A}^{(i-1)}$ then break
7: end for
8: return $\bar{\mathcal{A}} = \mathcal{A}^{(i)}$
```

Algorithm 3: RELEVANCE-F (Fact Relevance)

```
INPUT:   $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, G, cost \rangle$ // Planning task
OUTPUT:  $\bar{\mathcal{A}}$ // Relevant actions

1: $F^{(0)} = G$ // Initialize relevant facts with goal facts
2: $\mathcal{A}^{(0)} = \emptyset$ // Initialize relevant actions with empty set
3: for $i = 1, 2, \dots$ do
4:     $\mathcal{A}^{(i)} = \mathcal{A}^{(i-1)} \cup \{a \in \mathcal{A} \mid \mathit{eff}(a) \cap F^{(i-1)} \neq \emptyset\}$
5:     $F^{(i)} = F^{(i-1)} \cup \bigcup_{a \in \mathcal{A}^{(i)}} \mathit{pre}(a)$
6:     if $F^{(i)} = F^{(i-1)}$ and $\mathcal{A}^{(i)} = \mathcal{A}^{(i-1)}$ then break
7: end for
8: return $\bar{\mathcal{A}} = \mathcal{A}^{(i)}$
```

Our first observation is that instead of reasoning at the level of variables, we can reason at the level of *facts*, i.e. variable values. This leads to RELEVANCE-F (Algorithm 3), wherein we start by marking the goal facts as relevant and iteratively add actions whose effects contain relevant facts (Fig. 2, top right).[1] Both algorithms preserve all justified plans, but RELEVANCE-F often prunes more than RELEVANCE-V.

Theorem 2 *Given a task* Π*, let* $\bar{\mathcal{A}}_V = \text{RELEVANCE-V}(\Pi)$ *and* $\bar{\mathcal{A}}_F = \text{RELEVANCE-F}(\Pi)$*.* $\bar{\mathcal{A}}_V$ *and* $\bar{\mathcal{A}}_F$ *each contain all actions from all justified plans for* Π.

Proof:

We first consider RELEVANCE-F (Algorithm 3).

Recall that a justified plan is one where each action establishes either a goal fact or a precondition of a subsequent action. Let $\pi = \langle a_0, a_1, \dots, a_{n-1} \rangle$ be any such plan. Suppose at least one of the actions in π is not in $\bar{\mathcal{A}}_F$, and, without loss of generality, let a_j be the last such action.

All actions a_k, for $k > j$, are in $\bar{\mathcal{A}}_F$. So, in the final iteration i of the algorithm, $F^{(i-1)}$ contains their preconditions: $G \cup \bigcup_{k=j+1}^{n-1} \mathit{pre}(a_k) \subseteq F^{(i)} = F^{(i-1)}$, and $\mathit{eff}(a_j) \cap F^{(i-1)} = \emptyset$ (since $a_j \notin \bar{\mathcal{A}}_F$). But since a_j is justified for π, $\mathit{eff}(a_j) \cap (G \cup \bigcup_{k=j+1}^{n-1} \mathit{pre}(a_k)) \neq \emptyset$, which is a contradiction. Therefore, every action in π is in $\bar{\mathcal{A}}_F$, and this holds for any justified plan π.

Now note that for any set of facts F, we have that $(\mathit{eff}(a_i) \cap F \neq \emptyset) \implies \mathcal{V}(\mathit{eff}(a_i)) \cap \mathcal{V}(F) \neq \emptyset$. Whenever Algorithm 3 marks a fact relevant in $F^{(i)}$, Algorithm 2 marks *all* facts for the same variable relevant in $\mathcal{V}^{(i)}$. Thus, $\bar{\mathcal{A}}_F \subseteq \bar{\mathcal{A}}_V$, and RELEVANCE-V (Algorithm 2) preserves all actions on all justified plans. $\square$

Corollary 2 $\bar{\mathcal{A}}_V$ *and* $\bar{\mathcal{A}}_F$ *each contain all actions from all perfectly justified plans for* Π.

Corollary 3 $\bar{\mathcal{A}}_V$ *and* $\bar{\mathcal{A}}_F$ *each contain all actions from all shortest optimal plans for* Π.

[1]We essentially adapt Brafman's (2001) RELEVANT-K algorithm to FDR, for k=1, and characterize the actions it preserves.

The Power of Fact-Level Analysis

Our next observation is that the stopping criteria can be refined. Since we already track which actions are relevant, we can check whether any relevant action threatens a fact that appears in the initial state. An initial-state fact that is unthreatened by any relevant action is said to be *causally linked* to the initial state.[2] We proceed as before, but now we filter out these (currently) causally linked facts prior to computing relevant actions (see Algorithm 4). Next, we add the preconditions of these relevant actions to our list of *unfiltered* relevant facts.[3] Then, we filter again (based on the new set of relevant actions), and repeat until we reach a fixed point. We can thus stop the iteration early whenever the only reason to mark further actions relevant would be due to them achieving causally linked facts.

We show an example of this in Figure 2 (middle-right). Suppose the task has goal `has-shears` and initial state `has-stone-pickaxe`. Since the only precondition fact of `mine-iron` is `has-stone-pickaxe`, and neither `mine-iron` nor `craft-shears` threatens that fact, we need not mark as relevant `craft-stone-pickaxe`, nor any of its ancestors.

Theorem 3 *Given a task* Π*,* $\bar{\mathcal{A}}_C = \text{RELEVANCE-FC}(\Pi)$ *contains all actions from all perfectly justified plans for* Π.

Proof:

Suppose $\pi = \langle a_0, a_1, \dots, a_{n-1} \rangle$ is a perfectly justified plan for task Π. Suppose at least one of the actions in π is not in $\bar{\mathcal{A}}_C$, and, without loss of generality, let a_j be the last such action.

All actions a_k, for $k > j$, are in $\bar{\mathcal{A}}_C$. So, in the final iteration i of the algorithm, $F^{(i-1)}$ contains their preconditions: $G \cup \bigcup_{k=j+1}^{n-1} \mathit{pre}(a_k) \subseteq F^{(i)} = F^{(i-1)}$. Additionally, $\mathit{eff}(a_j) \cap \bar{F}^{(i-1)} = \emptyset$ (since $a_j \notin \mathcal{A}^{(i)}$).

There are two possible explanations for this: either $\mathit{eff}(a_j) \cap F^{(i-1)} = \emptyset$, or $\mathit{eff}(a_j) \cap F^{(i-1)} \neq \emptyset$ and

[2]This is inspired by McAllester and Rosenblitt (1991), which describes causal links between actions.

[3]Storing the unfiltered relevant facts ensures that we still consider facts that were previously causally linked but that may have become threatened by a relevant action in a later iteration.

$(\mathit{eff}(a_j) \cap F^{(i-1)}) \subseteq C^{(i)}$. The former means no fact in a_j's effect appears in the goal or any subsequent precondition. This is a contradiction as a_j is part of a perfectly justified plan.

The latter means any fact f in a_j's effect that does appear in the goal or a subsequent precondition was already established by the initial state, and its value is unthreatened by any action in $\mathcal{A}^{(i)}$. This follows from the definition of $C^{(i)}$, but note that f may still be threatened by an action *not* in $\mathcal{A}^{(i)}$ (and a_j restores it). Therefore, if we delete a_j from the action sequence, the only way it could invalidate the plan is if a previous action $a'_j \notin \mathcal{A}^{(i)}$ in the plan threatens a causally linked fact.

Repeat the above analysis for the penultimate action not in $\mathcal{A}^{(i)}$, which now serves the same role as a_j. Again, it can be deleted. As before, the only way this could invalidate the plan is if a previous action $a''_j \notin \mathcal{A}^{(i)}$ in the plan threatens a causally linked fact. Delete all the remaining actions not in $\mathcal{A}^{(i)}$, one by one, until none remain. At this point, the remaining sequence must be a plan, since no remaining actions threaten the causally linked facts, and none of the removed actions achieved any subsequent precondition or goal facts. (This is true even if the *initial* action in the plan was deleted, thus the sequence remains applicable in the initial state.) It is also a plan reduction, since it is shorter than π, which contradicts the fact that π was perfectly justified. □

Corollary 4 *$\bar{\mathcal{A}}_C$ contains all actions from all shortest optimal plans for* Π.

Action Merging

Reasoning at the fact-level also allows us to perform the relevance analysis in a different labeled transition system.

Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, G, cost\rangle$ be a planning task; let $\bar{\mathcal{A}} \subseteq \mathcal{A}$ be a set of relevant actions; and let $\bar{\mathcal{V}} \subseteq \mathcal{V}$ be a set of relevant variables. Partition the actions in $\bar{\mathcal{A}}$ by cost and effects on variables in $\bar{\mathcal{V}}$. For each partition $\bar{\mathcal{A}}_k$, merge the actions to form a new action $\bar{a}_k$, as follows. For the precondition, compute the disjunction of the individual preconditions (DNF) and simplify the result: $Pre(\bar{a}_k) = \textsc{SimplifyDNF}(\bigvee_{a\in\mathcal{A}_k} pre(a))$. The aim of this process is to produce a DNF with fewer facts, though the resulting DNF may still include multiple disjunctions, which we then represent by separate actions. For the cost and effect, take the (common) cost and effect on variables in $\bar{\mathcal{V}}$ (guaranteed by the partition). Let $\mathcal{A}'$ be the set of these merged actions, and define $\Pi' = \langle \mathcal{V}, \mathcal{A}', s_0, G, cost\rangle$. The labeled transition system of Π' can be obtained from the labeled transition system of Π via label reduction (Helmert, Haslum, and Hoffmann 2007; Sievers, Wehrle, and Helmert 2014).

Provided that only variables in $\bar{\mathcal{V}}$ appear in the merged preconditions, the labeled transition system only differs in its labels and its effects on variables outside of $\bar{\mathcal{V}}$. Reachability (backward and forward) of facts about variables in $\bar{\mathcal{V}}$ remains the same. When $\bar{\mathcal{V}}$ is taken to be the set of (currently) relevant variables, this means we can alternatively perform relevance analysis in the label reduced transition system. In Alg. 5, the

Algorithm 4: Relevance-FC (Causal Links)

```
INPUT:   $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, G, cost\rangle$ // Planning task
OUTPUT:  $\bar{\mathcal{A}}$ // Relevant actions
1:  $F^{(0)} = G$ // Initialize relevant facts with goal facts
2:  $\mathcal{A}^{(0)} = \emptyset$ // Initialize relevant actions with empty set
3:  for $i = 1, 2, \ldots$ do
4:      $T = (\bigcup_{a\in\mathcal{A}^{(i-1)}} \mathit{eff}(a)) \setminus s_0$ // Threatening facts
5:      $C^{(i)} = s_0[\mathcal{V} \setminus \mathcal{V}(T)]$ // Causally linked facts
6:      $\bar{F}^{(i)} = F^{(i-1)} \setminus C^{(i)}$ // Filter out causally linked facts
7:      $\mathcal{A}^{(i)} = \mathcal{A}^{(i-1)} \cup \{a \in \mathcal{A} \mid \mathit{eff}(a) \cap \bar{F}^{(i)} \neq \emptyset\}$
8:      $F^{(i)} = F^{(i-1)} \cup \bigcup_{a\in\mathcal{A}^{(i)}} pre(a)$
9:      if $F^{(i)} = F^{(i-1)}$ and $\mathcal{A}^{(i)} = \mathcal{A}^{(i-1)}$ then break
10: end for
11: return $\bar{\mathcal{A}} = \mathcal{A}^{(i)}$
```

relevance analysis considers the preconditions of $\mathcal{A}'$, rather than $\mathcal{A}^{(i)}$, when marking facts relevant. While the relevant variables and values in these labeled transition systems are the same, their sets of plans might differ and removing irrelevant facts may result in removing a perfectly justified plan.

For example, in the Minecraft domain (Fig. 2, bottom-right), the `gather` action has precondition `hungry`, while `hunt` has no precondition and achieves the same (relevant) fact,[4] `tribe-has-food`, for the same cost. Merging `gather` and `hunt` simplifies the preconditions to those of `hunt` (i.e. empty), so `wait` (which achieves `hungry`) is not marked relevant. If the task has goal `tribe-has-food` and initial state ¬`hungry`, then `[wait, gather]` is a perfectly justified plan, and `[hunt]` is a shortest optimal plan. Relevance-FCM (Alg. 5) removes `wait` and the perfectly justified plan, but not the optimal plan.

Theorem 4 *Given a task* Π, $\bar{\mathcal{A}}_M = \textsc{Relevance-FCM}(\Pi)$ *contains all actions from all shortest optimal plans for* Π.

Proof:

Suppose $\pi = \langle a_0, a_1, \ldots, a_{n-1}\rangle$ is a shortest optimal plan for task Π, with corresponding state sequence $\langle s_0, s_1, \ldots, s_n\rangle$. Suppose at least one of the actions in π is not in $\bar{\mathcal{A}}_M$, and, without loss of generality, let a_j be the last such action.

All actions a_k, for $k > j$, are in $\bar{\mathcal{A}}_M$. So, in the final iteration i of the algorithm, $F^{(i-1)}$ contains their (possibly reduced and simplified) preconditions. If no partition contains more than one action, or if no partition's preconditions simplify, then $\bigcup_{a\in\mathcal{A}'} pre(a) = \bigcup_{a\in\mathcal{A}^{(i)}} pre(a)$ and Theorem 3 applies. Otherwise we must consider what happens during the label reduction and precondition simplification from $\mathcal{A}^{(i)}$ to $\mathcal{A}'$.

Suppose the partition computed by Reduce contains an action set $\bar{\mathcal{A}}_p \subseteq \mathcal{A}^{(i)}$ with at least two actions whose preconditions simplify to form a merged action set $\bar{\mathcal{A}}'_p$. Specifically, if $P = \bigcup_{a\in\bar{\mathcal{A}}_p} pre(a)$ and

[4] `hunt` also achieves `hungry`, but it is not task-relevant.

Algorithm 5: RELEVANCE-FCM (Casual Links & Merging)

```
INPUT:   $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, G, cost \rangle$ // Planning task
OUTPUT:  $\bar{\mathcal{A}}$ // Relevant actions

1:  $F^{(0)} = G$ // Facts from goal state
2:  $\mathcal{A}^{(0)} = \emptyset$ // Initialize relevant actions
3:  for $i = 1, 2, \ldots$ do
4:      $T = (\bigcup_{a \in \mathcal{A}^{(i-1)}} \mathit{eff}(a)) \setminus s_0$ // Threatening facts
5:      $C^{(i)} = s_0[\mathcal{V} \setminus \mathcal{V}(T)]$ // Causally linked facts
6:      $\bar{F}^{(i)} = F^{(i-1)} \setminus C^{(i)}$ // Filter out causally linked facts
7:      $\mathcal{A}^{(i)} = \mathcal{A}^{(i-1)} \cup \{a \in \mathcal{A} \mid \mathit{eff}(a) \cap \bar{F}^{(i)} \neq \emptyset\}$
8:      $\mathcal{A}' = \text{REDUCE}(\Pi, \mathcal{A}^{(i)}, \mathcal{V}(F^{(i-1)}))$
9:      $F^{(i)} = F^{(i-1)} \cup \bigcup_{a \in \mathcal{A}'} pre(a)$
10:     if $F^{(i)} = F^{(i-1)}$ and $\mathcal{A}^{(i)} = \mathcal{A}^{(i-1)}$ then break
11: end for
12: return $\bar{\mathcal{A}} = \mathcal{A}^{(i)}$
```

$P' = \bigcup_{a \in \bar{\mathcal{A}}'_p} pre(a)$, then $P' \subset P$.

We are interested in the case where some fact $f \in \mathit{eff}(a_j)$ is not added to $F^{(i)}$ *because of* the merging process—that is, $f \in P \setminus P'$. If no such fact exists for any partition, then we can delete a_j from the plan using the same reasoning as in Theorem 3. Otherwise, let $v = \mathcal{V}(f)$, and notice that since $f \notin P'$, *all* values for v must appear in P, i.e. $\{\langle v, d \rangle \mid d \in \mathcal{D}(v)\} \subseteq P$. Thus, a_j achieves fact f in a precondition of some subsequent action a_f, but for each other value of v, there exists another action a'_f (at least one, but possibly more), whose cost and effect are the same and whose precondition is applicable.

We can delete a_j from the plan and replace a_f with the a'_f that is applicable in state s_{j-1}, i.e. $pre(a'_f)[v] = s_{j-1}[v]$. The consequence is that we can construct an even shorter plan than π of equal or lesser cost, which is a contradiction, since π is already shortest optimal. □

Reachability and Relevance Iteration

The backwards goal-relevance analysis of Algorithm 5 may remove some facts or actions from the task that cause other facts or actions to become unreachable from the initial state in the pruned task. Thus, we can potentially achieve additional task pruning by running forward reachability analysis (Blum and Furst 1997; Hoffmann and Nebel 2001a; Alcázar and Torralba 2015) on the pruned task (see Algorithm 7).

Theorem 5 *Given a task* Π, $\bar{\mathcal{A}}_R = \text{REACHABILITY}(\Pi)$ *contains all actions from all plans for* Π.

Proof:

At each iteration of the algorithm, we mark as reachable all actions whose preconditions are applicable for the (currently) reachable facts. We then mark all reachable actions' effects as reachable facts. All actions on plans are applicable, so this will retain them all. □

Algorithm 6: REDUCE

```
INPUT:   $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, G, cost \rangle$ // Planning task
         $\bar{\mathcal{A}} \subseteq \mathcal{A}$ // Subset of actions
         $\bar{\mathcal{V}} \subseteq \mathcal{V}$ // Subset of variables
OUTPUT:  $\bar{\mathcal{A}}'$ // Merged actions

1: $\mathcal{P} \leftarrow$ Partition $\bar{\mathcal{A}}$ by cost and effects on $\bar{\mathcal{V}}$.
2: Initialize $\bar{\mathcal{A}}' = \emptyset$.
3: for all $\langle \bar{\mathcal{A}}_k, \mathit{eff}_k, cost_k \rangle \in \mathcal{P}$ do
4:     $Pre(\bar{a}'_k) = \text{SIMPLIFYDNF}(\bigvee_{a \in \bar{\mathcal{A}}_k} pre(a))$
5:     $\bar{\mathcal{A}}'_k = \{\langle pre(\bar{a}'_k), \mathit{eff}_k, cost_k \rangle \mid pre(\bar{a}'_k) \in Pre(\bar{a}'_k)\}$
6:     $\bar{\mathcal{A}}' = \bar{\mathcal{A}}' \cup \bar{\mathcal{A}}'_k$
7: end for
8: return $\bar{\mathcal{A}}'$
```

Algorithm 7: REACHABILITY

```
INPUT:   $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, G, cost \rangle$ // Planning task
OUTPUT:  $\bar{\mathcal{A}}$ // (Relaxed) reachable actions

1: $F^{(0)} = s_0$ // Initialize reachable facts with initial facts
2: $\mathcal{A}^{(0)} = \emptyset$ // Initialize reachable actions with empty set
3: for $i = 1, 2, \ldots$ do
4:     $\mathcal{A}^{(i)} = \mathcal{A}^{(i-1)} \cup \{a \in \mathcal{A} \mid pre(a) \subseteq F^{(i-1)}\}$
5:     $F^{(i)} = F^{(i-1)} \cup \bigcup_{a \in \mathcal{A}^{(i)}} \mathit{eff}(a)$
6:     if $F^{(i)} = F^{(i-1)}$ and $\mathcal{A}^{(i)} = \mathcal{A}^{(i-1)}$ then break
7: end for
8: return $\bar{\mathcal{A}} = \mathcal{A}^{(i)}$
```

Algorithm 8: RELEVANCEITERATION

```
INPUT:   $\Pi$ // Planning task
OUTPUT:  $\Pi'$ // Pruned task

1: loop
2:     $\Pi' = \text{PRUNE}(\Pi, \text{RELEVANCE}(\Pi))$ // e.g. Algs. 2–5
3:     $\Pi' = \text{PRUNE}(\Pi', \text{REACHABILITY}(\Pi'))$ // e.g. Alg. 7

4:     if $\Pi' = \Pi$ then break
5:     $\Pi = \Pi'$
6: end loop
7: return $\Pi'$
```

Finally, running Algorithm 7 after Algorithm 5 may once again remove actions or facts from the task and cause new facts or actions to become irrelevant, making further pruning possible. We therefore introduce one last algorithmic improvement, which is to repeatedly interleave relevance and reachability pruning (see Algorithm 8).

Theorem 6 *Given a task* Π, RELEVANCEITERATION, *combined with* RELEVANCE-$\{$V, F, FC$\}$, *preserves all perfectly justified plans, with* RELEVANCE-FCM, *preserves all shortest optimal plans, and never introduces any new plans.*

Proof:

Follows directly from Theorems 1, 2, 3, 4, and 5. □

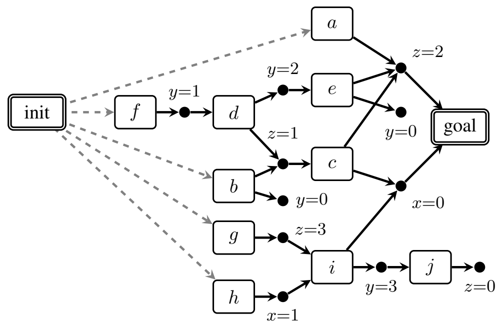
IPC (all 65 domains)	None	FD	V	F	FC	FCM	FCMR	FCMRL
Actions - Sum	17001557	16699722	16699722	16141206	16128063	16128035	16009673	16009340
Facts - Sum	1138370	858628	858628	854767	852463	852463	849761	849733
Variables - Sum	363961	237096	237096	235923	235660	235660	234309	234295
Coverage - Sum	970	982	981	990	989	988	986	986
Pruning (s) - Geo. mean	None	None	0.07	0.07	0.07	0.09	0.11	0.11
Translation (s) - Geo. mean	0.18	0.18	0.25	0.25	0.25	0.28	0.29	0.30
Search (s) - Geo. mean	2.09	1.83	1.82	1.70	1.67	1.68	1.67	1.68
Total (s) - Geo. mean	2.20	1.92	1.91	1.78	1.76	1.76	1.73	1.74

IPC (20 improved domains)	None	FD	V	F	FC	FCM	FCMR	FCMRL
Actions - Sum	4898612	4679923	4679923	4121407	4108264	4108236	3989874	3989541
Facts - Sum	682666	426090	426090	422229	419925	419925	417223	417195
Variables - Sum	234224	118942	118942	117769	117506	117506	116155	116141
Task Size - Sum	40320929	36693077	36567962	34117422	34011938	34011802	32954966	32952845
Coverage - Sum	283	289	289	298	297	296	296	295
Pruning (s) - Sum	None	None	517.04	532.31	599.50	3892.72	4556.40	5068.87
Translation (s) - Sum	1039.34	1042.41	1460.95	1487.78	1521.78	3686.79	4306.96	6092.84
Search (s) - Sum	23286.24	18633.88	18924.82	13263.46	12987.00	13048.43	13390.98	13243.94
Total (s) - Sum	23292.94	18640.26	18931.01	13269.79	12993.06	13054.59	13396.53	13249.37

Table 1: Results for IPC benchmark tasks. Fact-based relevance analysis results in additional pruning. (Top) All domains. (Bottom) Subset of domains where F+ pruning improves over V in some task.

Toy Example

For an example where each successive type of pruning further reduces task size, consider the following task.



Boxed nodes are actions, dots are facts, incoming edges denote preconditions, outgoing edges denote effects, and dashed edges indicate which actions can be executed from the initial state (all with empty preconditions). The initial state is $\{x{=}0; y{=}0; z{=}0\}$, and the goal is $\{x{=}0; z{=}2\}$.

Each pruning algorithm builds on the last: RELEVANCE-V (which we denote as V) considers variables;[5] F, FC, and FCM consider facts, causal links, and action merging, respectively; FCMR runs an extra REACHABILITY step; and FCMRL runs FCMR in a loop (i.e. RELEVANCEITERATION). On this task, V preserves j, since the goal mentions z, whereas F prunes j. FC finds a causal link from $x{=}0$ to the initial state and prunes $\{g, h, i\}$. FCM merges a and e, preventing $y{=}2$ from becoming relevant, which allows b and d to merge and f to be pruned. This causes d and e to become unreachable, so FCMR prunes them. Whenever we call PRUNE, we also delete any variables with <2 values; without h and i, this prunes x. This means FCMRL will merge a and c and prune b on the second relevance pass, and then prune c on the second reachability pass, leaving only a, which is the optimal plan.

[5]Technically, we should write PRUNE ∘ RELEVANCE-V, but we omit the PRUNE and function composition for brevity.

Experiments

We perform experiments on a wide range of benchmark planning tasks from the International Planning Competition (McDermott 2000; Long and Fox 2003; Coles et al. 2012; Vallati et al. 2015; Torralba and Pommerening 2018) and from our newly designed Minecraft domain (Figure 2).[6] The Minecraft domain and problem files are available at our code repository, along with the implementations of the pruning algorithms.

We implement our algorithms in the Fast Downward planning system (v23.06) (Helmert 2006) as a new step after translation but prior to search. The default behavior of the translator module runs the causal-graph-based variable relevance analysis, converts to FDR, and then runs REACHABILITY. We include this behavior as a baseline (denoted FD in Table 1), and show it is equivalent to disabling causal graph analysis for FDR conversion (`--keep-unimportant-variables`) and running REACHABILITY followed by RELEVANCE-V (denoted V). We implement each algorithm: V, F, FC, FCM, FCMR, and FCMRL. The NONE baseline performs no relevance pruning prior to search. All methods subsequently search for optimal plans with `astar(lmcut())`, since the pruning methods we consider preserve shortest optimal plans.

We measure the effectiveness of the pruning and the sub-

[6]All experiments used identical hardware (AMD Ryzen 9 7900X3D CPU, 250 GB of RAM). The IPC experiments used a search budget of 30 minutes, and the Minecraft ones used 5 minutes.

Size	**Coverage**			**Actions**			**Facts**			**Variables**			**Prune (s)**	**Total (s)**	
	FD	F[†]	FCM	FD	F[†]	FCM	FD	F[†]	FCM	FD	F[†]	FCM	FCM	FD	FCM
1	10	10	10	22	16	6	31	26	8	15	13	4	0.01	0.01	0.01
2	10	10	10	67	48	12	55	48	10	28	24	5	0.01	54.48	0.01
5	0	0	10	587	410	30	127	127	16	64	64	8	0.01	-	0.01
10	0	0	10	4,177	2,777	60	251	251	26	126	126	13	0.02	-	0.02
12	0	0	10	7,125	4,578	72	300	300	30	150	150	15	0.05	-	0.10
15	0	0	10	13,764	8,776	90	369	369	36	184	184	18	0.08	-	0.97
20	0	0	10	32,349	20,449	120	485	485	46	243	243	23	0.21	-	44.14
25	0	0	0	62,938	40,101	150	609	609	56	304	304	28	0.55	-	-

Table 2: Results for 80 generated Minecraft tasks, varying task size (i.e. number of agents). Fact-based relevance analysis reduces actions by up to 99%, facts and variables by up to 90%, and solves 50 more problems than the baseline Fast Downward algorithm (FD). The † symbol indicates that results for FC are the same as F.

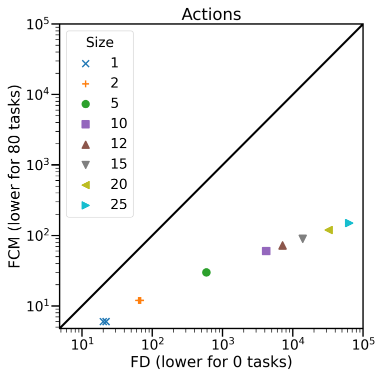


Figure 3: Number of actions remaining after pruning for FCM and FD.

sequent search. In Table 1, we report the numbers of actions, facts, and variables after pruning, along with solution coverage and time. We include results for all 65 domains, as well as the subset of 20 domains where fact-level analysis (F+) reduces task size relative to variable-level (V) for at least one task. We find that fact-level pruning reduces total task size and improves coverage, without increasing total time.

Table 2 shows pruning performance for the Minecraft domain as the number of agents increases, for FD, F, FC, and FCM. The latter reduces actions by up to 99%, facts and variables by up to 90%, and solves 50 more problems than the baseline Fast Downward algorithm. We plot the relative pruning performance (number of actions) for FD and FCM in Figure 3. Relative improvement increases with task size.

Related Work

The idea of using goal information to guide the search process dates back to the earliest planners (Newell, Shaw, and Simon 1959; Fikes and Nilsson 1971), which used the goal mainly for means-ends analysis. Subsequent work introduced "goal regression"—starting from the goal and applying actions in reverse until reaching the initial state (Warren 1974; Waldinger 1975). Such approaches are similar in spirit to ours, but did not remove irrelevant information prior to search.

In the context of pruning irrelevant information, Nebel, Dimopoulos, and Koehler (1997) pointed out that the initial state and goal constitute two crucial sources of guidance: if an action or fact is unreachable by forward-chaining from the initial state or backward-chaining from the goal, then it cannot be part of any solution. They further proved that fully removing all irrelevant information is as hard as planning itself (i.e. PSPACE-hard), and while their work included a polynomial-time approximation algorithm, it was not always solution preserving. Boutilier, Brafman, and Geib (1998) and Brafman (2001) respectively introduced the Reachable-k and Relevance-k algorithms, which *were* solution preserving; however, increasing amounts of pruning (due to increasing k) came with an exponential increase in computational complexity. Hoffmann and Nebel (2001b) instead considered pruning a *relaxed* version of the task, which allowed them to safely improve the planner's heuristic computation, but they did not prune the task itself. Haslum, Helmert, and Jonssson (2013) prune tasks while retaining at least one optimal plan by searching for replaceable paths through the domain transition graph, which is efficient but orthogonal to our approach.

Modern planners try to strike a balance between pruning cost and planning cost. The Fast Downward planning system (Helmert 2006) performs causal graph analysis of state variables to approximate which variables are goal-relevant and then uses relaxed reachability analysis to ignore actions that are unreachable from the initial state. We prove that this process is guaranteed to preserve all actions from justified plans, but our experiments show it is also far from optimal.

Conclusion

We introduce several algorithms for pruning irrelevant task information from planning tasks. Our algorithms consider relevance at the level of facts, rather than variables, which enables much more aggressive pruning. The algorithms iterate until they reach a fixed point, and fact-based relevance analysis allows us to terminate this iteration early—either because precondition facts are causally linked to the initial state, or because actions can be merged together with simplified preconditions. We formally characterize the set of plans preserved by each algorithm we introduce, and in particular show that they all preserve shortest optimal plans without introducing new ones. We test our algorithms on a wide range of planning tasks, and show that the approach can significantly reduce problem size and planning time when tasks contain irrelevant information.

Acknowledgements

We are deeply grateful to Michael Fishman, Nishanth Kumar, Natasha Danas, Michael Littman, and Stefanie Tellex for their efforts in getting the original version of this project started (Fishman et al. 2023) and for encouraging us to keep going. Thanks also to Jörg Hoffmann, Stuart Russell, and Naman Shah, for helpful discussions, as well as to our reviewers, for their valuable comments and for helping us fix an error in one of the proofs.

This research was supported by a gift from Open Philanthropy to the Center for Human-Compatible Artificial Intelligence at UC Berkeley, an AI2050 Senior Fellowship for Stuart Russell from the Schmidt Fund for Strategic Innovation, and by the Office of Naval Research (ONR) under REPRISM MURI N000142412603 and ONR grant N00014-22-12592.

References

Alcázar, V.; and Torralba, Á. 2015. A Reminder about the Importance of Computing and Exploiting Invariants in Planning. In Brafman, R.; Domshlak, C.; Haslum, P.; and Zilberstein, S., eds., *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2015)*, 2–6. AAAI Press.

Bäckström, C.; and Nebel, B. 1995. Complexity Results for SAS$^+$ Planning. *Computational Intelligence*, 11(4): 625–655.

Blum, A.; and Furst, M. L. 1997. Fast Planning Through Planning Graph Analysis. *Artificial Intelligence*, 90(1–2): 281–300.

Boutilier, C.; Brafman, R. I.; and Geib, C. 1998. Structured Reachability Analysis for Markov Decision Processes. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI 1998)*, 24–32.

Brafman, R. 2001. On Reachability, Relevance, and Resolution in the Planning as Satisfiability Approach. *Journal of Artificial Intelligence Research*, 14: 1–28.

Coles, A.; Coles, A.; García Olaya, A.; Jiménez, S.; Linares López, C.; Sanner, S.; and Yoon, S. 2012. A Survey of the Seventh International Planning Competition. *AI Magazine*, 33(1): 83–88.

Fikes, R. E.; and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2: 189–208.

Fink, E.; and Yang, Q. 1992. Formalizing Plan Justifications. In *Proceedings of the 9th Canadian Conference on Artificial Intelligence (CAI 1992)*.

Fink, E.; and Yang, Q. 1993. A Spectrum of Plan Justifications. In *Proceedings of the AAAI 1993 Spring Symposium*, 29–33. AAAI Press/MIT Press.

Fishman, M.; Kumar, N.; Allen, C.; Danas, N.; Littman, M.; Tellex, S.; and Konidaris, G. 2023. Task Scoping: Generating Task-Specific Simplifications of Open-Scope Planning Problems. In *IJCAI 2023 Workshop on Bridging the Gap Between AI Planning and Reinforcement Learning*.

Haslum, P.; Helmert, M.; and Jonssson, A. 2013. Safe, Strong and Tractable Relevance Analysis for Planning. In Borrajo, D.; Kambhampati, S.; Oddi, A.; and Fratini, S., eds., *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 317–321. AAAI Press.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Helmert, M. 2009. Concise Finite-Domain Representations for PDDL Planning Tasks. *Artificial Intelligence*, 173: 503–535.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible Abstraction Heuristics for Optimal Sequential Planning. In Boddy, M.; Fox, M.; and Thiébaux, S., eds., *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS 2007)*, 176–183. AAAI Press.

Hoffmann, J.; and Nebel, B. 2001a. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, 14: 253–302.

Hoffmann, J.; and Nebel, B. 2001b. RIFO Revisited: Detecting Relaxed Irrelevance. In Cesta, A.; and Borrajo, D., eds., *Proceedings of the Sixth European Conference on Planning (ECP 2001)*, 127–135. AAAI Press.

Katz, M.; Röger, G.; and Helmert, M. 2022. On Producing Shortest Cost-Optimal Plans. In Chrpa, L.; and Saetti, A., eds., *Proceedings of the 15th Annual Symposium on Combinatorial Search (SoCS 2022)*, 100–108. AAAI Press.

Long, D.; and Fox, M. 2003. The 3rd International Planning Competition: Results and Analysis. *Journal of Artificial Intelligence Research*, 20: 1–59.

McAllester, D.; and Rosenblitt, D. 1991. Systematic Nonlinear Planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI 1991)*, 634–639. AAAI Press/MIT Press.

McDermott, D. 2000. The 1998 AI Planning Systems Competition. *AI Magazine*, 21(2): 35–55.

Nebel, B.; Dimopoulos, Y.; and Koehler, J. 1997. Ignoring Irrelevant Facts and Operators in Plan Generation. In Steel, S.; and Alami, R., eds., *Recent Advances in AI Planning. 4th European Conference on Planning (ECP 1997)*, volume 1348 of *Lecture Notes in Artificial Intelligence*, 338–350. Springer-Verlag.

Newell, A.; Shaw, J. C.; and Simon, H. A. 1959. Report on General Problem-Solving Program. In *Proceedings of the 1st International Conference on Information Processing (IFIP 1959)*, 256–264.

Salerno, M.; Fuentetaja, R.; and Seipp, J. 2023. Eliminating Redundant Actions from Plans using Classical Planning. In Marquis, P.; Son, T. C.; and Kern-Isberner, G., eds., *Proceedings of the Twentieth International Conference on Principles of Knowledge Representation and Reasoning (KR 2023)*, 774–778. IJCAI Organization.

Sievers, S.; Wehrle, M.; and Helmert, M. 2014. Generalized Label Reduction for Merge-and-Shrink Heuristics. In Brodley, C. E.; and Stone, P., eds., *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2014)*, 2358–2366. AAAI Press.

Torralba, Á.; and Pommerening, F. 2018. IPC 2018—Classical Tracks. In *The International Planning Competition 2018*.

Vallati, M.; Chrpa, L.; Grześ, M.; McCluskey, T. L.; Roberts, M.; and Sanner, S. 2015. The 2014 International Planning Competition: Progress and Trends. *AI Magazine*, 36(3): 90–98.

Waldinger, R. 1975. Achieving several goals simultaneously. Technical Report 107, Stanford Research Institute.

Warren, D. H. 1974. WARPLAN: A system for generating plans. Technical Report 76, University of Edinburgh.