Filtering Top-k Relevant Plans

Mauricio Salerno, Miguel Tabernero, Raquel Fuentetaja, Alberto Pozanco

Department of Computer Science and Engineering
Universidad Carlos III de Madrid, 28911 Leganés, Madrid, Spain
msalerno@pa.uc3m.es, mtaberne@pa.uc3m.es, rfuentet@inf.uc3m.es, alberto.pozanco@gmail.com

Abstract

The automatic generation of a set of plans rather than just one is a relevant problem in Automated Planning, with a wide range of applications, including applications to finance and banking. Such sets can be computed through top-k planning, which aims to find the best *k* plans that solve a planning task. Existing approaches to solve the top-k planning problem might generate plans that are not *relevant* for some practical applications. In particular, plans might contain actions that can be removed from the plan while maintaining its validity. These unnecessary actions not only increase the cost of plans, but might particularly reduce the utility of top-k planning. In this work we propose an Automated Planning approach for identifying and eliminating redundant actions from plans, and show how to incorporate this method into top-k planning to guarantee that the generated plans do not contain redundant actions. We perform an empirical analysis to study the existence of redundant actions in plans in several benchmarks, and analyze how top-k planning methods are affected when forced to find plans without redundant actions.

Introduction

There exist many planning applications where it is necessary to compute a set of plans rather than only one. This is the case of tools where planning is supporting human decision makers, who are typically keen on exploring different alternatives and scenarios. Having a diverse set of plans (Srivastava et al. 2007; Roberts, Howe, and Ray 2014) at hand when making these decisions is crucial in many finance applications, that include but are not limited to: goal and plan recognition (Ramírez and Geffner 2009; Sohrabi, Riabov, and Udrea 2016) to predict customer goals in order to provide adequate services to them (Borrajo, Gopalakrishnan, and Potluru 2020; Borrajo and Veloso 2020; Borrajo, Veloso, and Shah 2020); planning approaches to predict stock market movements (Mund, Vallati, and McCluskey 2020); or cybersecurity (Boddy et al. 2005; Pozanco et al. 2021), where experts are interested in understanding a set of possible attacks to communication networks and protocols.

Top-k planning (Riabov, Sohrabi, and Udrea 2014; Katz et al. 2018; Speck, Mattmüller, and Nebel 2020) is one of the existing approaches to compute sets of plans, which aims to find the best k plans solving a planning task. In top-k planning, as in other Automated Planning problems, the quality of plans is measured using a cost function. However, there are additional notions of quality that can be considered. Quality can be also defined from the point of view of *plan relevance* or *justified plans* (Fink and Yang 1992). From a domain independent perspective, relevant plans can be understood as those that do not contain loops and or irrelevant/unnecessary actions (Fink and Yang 1992; Nebel, Dimopoulos, and Koehler 1997). From a domain specific point of view more subtle notions of relevancy could be considered.

Current planners guided by powerful heuristics will probably not include (many) irrelevant actions in the output plan, but this is not enough for top-k planning. Iterative approaches to top-k have shown promising results (Katz et al. 2018). However, resulting plans include more irrelevant information as more iterations are carried out, to the point that truly alternative plans can not be easily distinguished from those that were extended with unnecessary operators. This may have an important negative impact in applications. For instance, in goal and plan recognition, agents are usually assumed to follow optimal or at *least* sub-optimal plans to achieve their goals (Ramírez and Geffner 2010), but redundant actions do not contribute to achieve the goals to such task. Also, it is not useful to consider these plans in explainable AI, since they are of little interest for users.

As an illustrative example of the notion of plan relevance consider the planning task depicted in Figure 1, where a

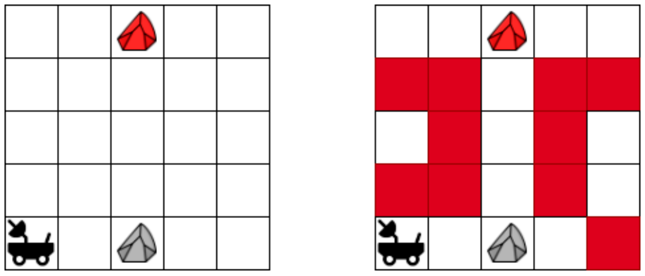
Figure 1: Two planning tasks of a navigation domain where a robot has to sample a rock. The robot can move to adjacent white tiles.

robot has to take a sample of the red rock situated on the top. Here, the plans that represent the different routes the agent can use to sample the red rock can be considered relevant of justified. In contrast, any plan where the agent passes through the same cell more than once would not be justified, since the actions that create the loop can be removed from the plan and it still achieves the goal. Another plan that would not be justified is one where the agent reaches the goal state (samples the rock) and then executes additional actions: these subsequent actions can be subtracted from the plan, and generate equally valid plans. The same occurs in plans where the robot samples also the grey rock, which is not necessary to sample. Figure 1 shows two planning problems. For the one on the left there are many relevant plans, while for the one on the right there is only one relevant plan. It is not difficult to find additional examples in large-scale real-world applications where the domain can include many operators, static facts and objects, which might be irrelevant for specific goals and initial situations. The number of possible non-relevant plans can be quite large, and these plans are somehow *artificial*, since they do not represent actual different alternatives to achieve the goals.

In this work, we focus on two ideas: (1) filtering unnecessary actions in a plan post-optimization step, in the same line as some previous works (Nakhost and Müller 2010; Chrpa, McCluskey, and Osborne 2012b,a; Balyo, Chrpa, and Kilani 2014); specifically, we contribute with a compilation that encodes the problem as an Automated Planning task, so that in can be solved using an off-the-shelf automated planner; and (2) incorporating the notion of plan justification in the context of top-k planning; specifically, we present an approach to find relevant/justified plans when using iterative approaches to top-k planning. This work expands on top-k and is closely related with the ideas of top-quality (Katz, Sohrabi, and Udrea 2020) and diverse planning (Katz and Sohrabi 2020), since these approaches also find plans that should meet certain conditions.

The rest of the paper is organized as follows. Next section introduces basic notions of classical planning, plan justification and top-k planning. Then, we define a planning compilation for plan justification. After that, we describe how to integrate it with top-k planning. Next, we include an empirical evaluation on how the incorporation of plan justification affects top-k planning. Finally, we discuss related work and draw some conclusions.

Background

Classical Planning

A classical planning task (Fikes and Nilsson 1971) is defined as a tuple $\Pi = (\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$, where $\mathcal{F}$ is a set of propositions; $\mathcal{A}$ is a set of actions; $\mathcal{I} \subseteq \mathcal{F}$ is the initial situation, encoding what propositions are true initially; and $\mathcal{G} \subseteq \mathcal{F}$ is a set of goal propositions. Every $a \in \mathcal{A}$ has preconditions, denoted as $pre(a) \subseteq \mathcal{F}$, added effects $add(a) \subseteq \mathcal{F}$ and negative effects $del(a) \subseteq \mathcal{F}$.

A planning task Π defines a state model which states $s \in S$ are subsets of $\mathcal{F}$ and are represented by the fluents that are true in the corresponding state. In this model, the initial state is $s_i = \mathcal{I}$, and the goal states are those s_g that include the goals $\mathcal{G} \subseteq s_g$. The actions $a \in \mathcal{A}$ that are applicable in a state s, denoted as $A(s)$, are those for which $pre(a) \subseteq s$. The transition function is γ, where $\gamma(s, a) = (s \setminus del(a)) \cup add(a)$ represents the state s' that results from the application of the action a in state s.

A solution or valid plan for Π is an action sequence $\pi = \langle a_1, \ldots, a_n \rangle$ that induces a state sequence $\mathcal{S}_\pi = \langle s_0, \ldots, s_n \rangle$ such that $s_0 = \mathcal{I}$ and, for each i such that $1 \leq i \leq n$, a_i is applicable in s_{i-1} and $s_i = \gamma(s_{i-1}, a_i$. A plan π solves Π if and only if $\mathcal{G} \subseteq s_n$. We denote the set of all the plans that solve a planning task Π as P_Π. Each action $a \in \mathcal{A}$ is assumed to have a non-negative cost $c(a)$, so that the cost of a plan is $c(\pi) = \sum c(a_i)$. A plan is optimal if it has minimum cost.

Top-k planning

The objective of a top-k planning problem (Riabov, Sohrabi, and Udrea 2014; Sohrabi, Riabov, and Udrea 2016) is to find the k plans of lowest cost for a planning task.

Definition 1. *A top-k planning problem is a tuple* (Π, k)*, where* $\Pi = (\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ *and* $k \in \mathbb{N}$*. The goal is to find a set of plans* $P_k \subseteq P_\Pi$ *such that:*

- *For each* $\pi \in P_k$*, if there exists a plan* π' *with* $c(\pi') < c(\pi)$ *then* $\pi' \in P$.
- $|P_k| \leq |P_\Pi|$*, where* $|P_k| < k$ *implies* $P_k = P_\Pi$.

Note that the plans in P_k are not required to meet any condition other than being those with the k lowest costs. This means that algorithms that solve the top-k planning problem might find plans with actions or set of actions that are not *relevant/justified* under certain settings. The conditions a plan must meet to be *relevant* might be domain dependent and depend on the semantics of the planning task. Nevertheless, plan justification has been widely studied, with special interest in finding redundant actions that can be removed from the plan without affecting its validity. We want to introduce this notion into top-k planning.

Katz et al. (2018) proposed an iterative approach to solve the top-k planning problem. The main idea of their algorithm is to find additional solutions to a planning task by reformulating the original task. The reformulations *forbid* already found plans, so that previous plans will not be valid solutions, thus forcing the planner to find an alternate solution. The best *k* plans are found by iteratively solving and reformulating planning tasks. We are interested in this approach because it is straightforward to extend, including additional conditions the found plans must meet. When a solution is found, it can then be checked to verify if it meets a certain condition. If it does, this solution is counted as a *valid* plan, and the iterative procedure continues as normal. If it does not meet the condition, this solution is forbidden with the reformulation as usual, but the number of found plans so far is not increased.

Plan Justification

The notion of plan justification can be traced back to the early 1990s (Fink and Yang 1992). In that work, Fink and

Yang define different types of plan justifications: backward justification, well justification and perfect justification.

Given $\Pi = (\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$, a plan $\pi = (a_1, ..., a_n)$ and the set of causal links between them, an action $a_i \in \pi$ is *backward justified* if $\exists p \in add(a_i)$ such that $p \in \mathcal{G}$ or $\langle a_i, a_j, p \rangle$ is a causal link and a_j is backward justified[1]. The triple $\langle a_i, a_j, p \rangle$ forms a causal link if a_i adds p, p is a precondition of a_j, and p is neither added nor deleted by any action between a_i and a_j (Celorrio et al. 2013). Then, a_i is backward justified if it is causally related to the goals. A plan π is backward justified if all of its actions are backward justified. *Well justified* actions are those that can not be removed from the plan without affecting the applicability of other actions. *Perfectly-justified* plans are those for which no subset of actions can be removed from the plan without invalidating plan. In this paper we also apply this idea.

AP compilations for Action Elimination

This section introduces some formal definitions used in the rest of the paper, and explains the proposed AP compilations to eliminate unnecessary actions from plans.

Formal Definitions

We use the notion of *perfectly-justified* introduced by Fink and Yang. Thus, a plan is perfectly-justified if no actions can be *skipped* or eliminated while maintaining the plans' validity. We introduce it using the following definitions.

Definition 2 (Reduced Plan). *Given a plan $\pi = \langle a_1, \ldots, a_n \rangle$ for Π and a strict subset of its actions $A_\pi \subset \pi$, $A_\pi \neq \emptyset$, the reduced plan $\pi_{\setminus A_\pi}$ is the action sequence resulting from eliminating the actions $a_i \in A_\pi$ from π.*

Definition 3 (Well-justified action set). *A subset of plan actions $A_\pi \subseteq \pi$, $A_\pi \neq \emptyset$, is well justified if the corresponding reduced plan $\pi_{\setminus A_\pi}$ is not a valid plan for Π.*

The previous definition just extends the notion of well-justified actions to well-justified subsets of actions. [2]

Now, we consider a plan π to be perfectly-justified if all of its subsets of actions are well justified, i.e. all the plans reduced by those subsets are invalid, so that there is no way of reducing the plan while maintaining its validity.

Definition 4 (Perfectly-justified plan). *A plan π is perfectly-justified iff all non-empty strict subsets of its actions, $A_\pi \subset \pi$, are well-justified.*

Then, if there is at least a subset of actions which is not well-justified, the plan is not perfectly-justified. In that case will say the actions in that subset are **unnecessary**.

Given Π and a plan π, the task of finding the smallest perfectly-justified plan by eliminating actions from π is called Minimal Length Reduction (MLR) (Balyo, Chrpa, and Kilani 2014). Balyo, Chrpa, and Kilani also define the Minimal Reduction (MR) task. The aim of this task is to find a plan with the smallest possible cost by eliminating actions from π. In this paper we are particularly interested in MLR. Both tasks have been shown to be NP-complete (Fink and Yang 1992; Nakhost and Müller 2010). In the following section we propose a compilation of the MLR problem into an AP one so that it can be solved with an off-the-shelf planner.

[1]Causal links were called initially *establishments* (Fink and Yang 1992).

[2]This notion is also defined by Balyo, Chrpa, and Kilani [2014] as plan reductions.

Perfect Justification as Planning

The idea is to define a classical planning task that, given a planning task and a solution plan, can identify and eliminate sets of unnecessary actions from plans. The compilation consists of encoding the planning task in a way that allows to include any action occurring in the original plan after the last action that was previously included. We achieve this by creating an order relation between the actions in π. More formally, given the planning task Π and a solution plan π we define $\Pi^{order} = (\mathcal{F}', \mathcal{A}', \mathcal{I}', \mathcal{G})$ as follows:

- $\mathcal{F}' = \mathcal{F} \cup \mathcal{F}_{last} \cup \mathcal{F}_{order} \cup \mathcal{F}_{planact}$, where:
  - $\mathcal{F}_{last} = \{last_i \,|\, 0 \leq i \leq n\}$ facts represent the last position considered. There is a position (order in the sequence) for every action in the original plan plus an additional zero position,
  - $\mathcal{F}_{order} = \{order_{i,j} \,|\, 0 \leq i \leq n, i < j \leq n\}$ are static facts to encode that position i is before position j, and
  - $\mathcal{F}_{planact} = \{planact\_a_i \,|\, 1 \leq i \leq n\}$ are static facts to represent the action a appears in the plan π at position i.
- $\mathcal{A}' = \{a_{i,j} \,|\, a \in \mathcal{A}, 0 \leq i \leq n, i < j \leq n\}$, where there is an $a_{i,j}$ action for every action a in the original task and combination of positions i, j, defined as follows:

$$\begin{aligned} pre(a_{ij}) &= pre(a) \cup \\ &\quad \{last_i, order_{i,j}, planact\_a_j\} \\ add(a_{ij}) &= add(a) \cup \{last_j\} \\ del(a_{ij}) &= del(a) \cup \{last_i\} \end{aligned}$$

- $\mathcal{I}' = \mathcal{I} \cup \{last_0\} \cup \{order_{i,j} \,|\, 0 \leq i, j < n, i < j\} \cup \{planact\_a_i \,|\, a_i \in \pi\}$.

Actions $a_{i,j}$ in $\mathcal{A}'$ will only be applicable if there is an occurrence of action $a \in \mathcal{A}$ in the original plan π at position j. For that, facts of type $planact\_a_i$ are included in $\mathcal{I}'$, representing the plan π. There is one of such facts per plan action, indicating π contains an occurrence action $a \in \mathcal{A}$ at position i. The $last_i$ fact represent the position of the last included action. Thus, the preconditions of $a_{i,j}$ actions check that the action being included at position j, occurs in the original plan before the last included action with position i. The new initial state $\mathcal{I}'$ sets the relation order between the actions in π, where there is a fact $order_{0,j}$ for every position j in the plan, which allows the application of any action of the plan.

When Π^{order} is solved, the resulting plan will only contain the actions in the original plan that are necessary without altering the order. The plan that solves Π^{order} can be easily compiled-back to be a plan of the original task, just by removing the action parameters representing orders. The correspondence between the actions of both plans is one-to-one.

Definition 5. *Let π' be a valid plan of Π^{order}. Then, the compiled-back plan for Π is $\pi'' = \{a \,|\, a \in \pi \wedge a_{ij} \in \pi'\}$.*

This means that every a_{ij} action is replaced by its original action a.

Proposition 1. *The plan π'' obtained from any valid plan π' for Π^{order} is a valid plan for Π.*

Proof sketch. A plan π that induces the state sequence $\mathcal{S}_\pi = \langle s_0, \dots, s_n\rangle$ is valid if all of its actions are applicable in the state they are applied and $\mathcal{G} \subseteq s_n$. Since the goals of both Π and Π^{order} are the same, if π' is valid for Π^{order}, then it will also achieve all the goals in Π by definition. Since $pre(a) \subset pre(a_{i,j})$ for every $a_{i,j} \in \mathcal{A}'$, if $a_{i,j}$ is applicable in a state, then the corresponding action $a \in \mathcal{A}$ is also applicable in that state. We know that π' is valid, so, starting from $\mathcal{I}$ all of its actions are applicable. Since $\mathcal{I} \subset \mathcal{I}'$, all actions in π'' are in turn applicable starting from $\mathcal{I}$. The goals are met and all actions are applicable, so π'' is a valid plan for Π.

Theoretical Properties

This section shows the type of plans that are obtained from solving the compiled planning tasks. In particular, we show that the set of valid plans for the compilations is the set of valid reduced plans of the original task.

Let P_Π be the set of valid plans for a planning task Π. Two planning tasks Π, Π' are *equivalent* if they have the same sets of valid plans, $P_\Pi = P_{\Pi'}$. For two plans π^-, π we say that π^- is a subset of π, $\pi^- \subset \pi$ if π^- can be generated from eliminating actions from π. Since the compilation **only** allows for the execution of actions in the original plan or for their (implicit or explicit) elimination, the set of valid plans for Π^{order} contain exactly all the valid plans that can be generated from eliminating subsets of actions from the original task if their corresponding transformations as defined in 5 are considered. More formally:

Proposition 2. *Let Π be a planning task and $\pi = \langle a_1, ..., a_n\rangle$ a valid plan for Π. Let P_{order} be the set of compiled-back valid plans for Π^{order}. Then $P_{order} = \{\pi^- \subseteq \pi \,|\, \pi^- \in P_\Pi\}$.*

Proof sketch. We have to show that (i) any plan in P_{order} is in $\{\pi^- \subseteq \pi \,|\, \pi^- \in P_\Pi\}$, and that (ii) any plan in $\{\pi^- \subseteq \pi \,|\, \pi^- \in P_\Pi\}$ is in P_{order}:
(i) Let $\pi' = \langle b_1, ..., b_m\rangle \in P_{order}$ be any compiled-back valid plan for Π^{order}. Since all a_{ij} belonging to Π^{order} have in their precondition $planact\_a_j$, only actions in π can be in π'. It is trivial to show that only one $last_i$ proposition is true in each state. In $\mathcal{I}$ only $last_0$ is true. Because of Π^{order} encoding, any a_i for which $pre(a_i) \subseteq \mathcal{I}$ is applicable initially. Any of these actions delete $last_0$ and add $last_i$. An action a_j is now applicable only if $i < j$. Continuing this process until $last_n$ is true it is easy to see that π' can only have actions in π respecting their order, where some of them might be skipped. By Proposition 1, $\pi' \in P_\Pi$. Therefore, it is proven that $(\forall\, \pi', \pi' \in P_{order} \implies \pi' \subseteq \pi \wedge \pi' \in P_\Pi)$.
(ii) Let $\pi = \langle b_1, ..., b_m\rangle \in \{\pi^- \subset \pi \,|\, \pi^- \in P_\Pi\}$. Following similar reasoning we can prove that $(\forall \pi, \pi \in \{\pi^- \subseteq \pi \,|\, \pi^- \in P_\Pi\} \implies \pi \in P_{order})$. This is omitted for space reasons. Therefore $P_{order} = \{\pi^- \subseteq \pi \,|\, \pi^- \in P_\Pi\}$. □

The branching factor for solving Π^{order} can be as high as $|\pi| = n$, and $(\sum_{i=1}^{n-1} i)$ *order* propositions must be created. This might become an issue when the length of plans is particularly long. We have another compilation introducing additional *skip* actions that allow to omit single actions, in which the order relation is defined only for consecutive actions. In this way the branching factor is reduced to exactly 2 in each step: either the current action or the skip action can be applied. In this paper we only consider Π^{order}. But, we have observed that the impact of using skip actions instead is not significant in our specific experiments.

Top-k Relevant Plans

This section introduces the top-k relevant planning problem. The idea is simple: given a characteristic/condition we are interested on, finding the best *k* plans that meet that condition. We denote the set of all plans that solve a planning task and meet a condition ω as ${P_\Pi}^\omega \subseteq P_\Pi$. More formally:

Definition 6 (Top-k relevant planning problem). *A top-k relevant planning problem is a tuple $\langle \Pi, k, \omega\rangle$, where $\Pi = (\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ is a planning task, $k \in \mathbb{N}$ and $\omega : P_\Pi \mapsto$ {true,false}. The goal is to find a set of plans $P_k \subseteq {P_\Pi}^\omega$ such that:*

- *For each $\pi \in P_k$, if there exists a plan π' such that $c(\pi') < c(\pi) \wedge \omega(\pi')$ then $\pi' \in P_k$.*
- *$|P_k| \leq |{P_\Pi}^\omega|$, where $|P_k| < k$ implies $P_k = {P_\Pi}^\omega$.*

This means that we want to find the k plans of least cost that meet a condition ω. In this work we are particularly interested in plans that are perfectly justified. To do so, we extend Katz et al. (2018) iterative top-k planning approach as the Algorithm 1 shows. Initially P_k (solution set) is empty. Then, as long as k relevant plans have not been found, we repeat the following procedure. Get the next plan using an iterative top-k planning approach. If the problem is unsolvable, we return the found plans so far. Otherwise, we check if the plan π meets the condition ω. If it does, it is added to P_k. Finally, a new planning task is defined following the iterative top-k planning procedure to forbid plans. We leave out the details of the reformulation of the planning task since we are using exactly the same approach proposed by Katz et al. (2018).

An open identified issue with this approach is determining when to stop. In many instances, the number of plans that meet a condition might be lower than the number of desired plans k. When the plans must be perfectly justified, an example of this is easy to imagine. For the planning problem shown on the right of Figure 1 there is only one perfectly justified plan. Any other plan, including plans with loops, contains sets of actions that can be eliminated from the plan to get equally valid plans. Since there is an infinite number of plans (with loops) that solve the problem, the described approach would continue trying to find plans to reach k indefinitely. After finding the first relevant plan, a new (loopy or with unnecessary actions) plan will be found on each iteration. It will not be perfectly justified, and therefore it will not be added to the solution set. Then, if $k > 1$, the process will continue indefinitely, unless it is explicitly stopped using time or memory limits or memory is exhausted. A possible solution would be to incorporate the relevance check into the search process. Specifically, it can be performed when checking the goal condition so that the search can continue when the plan is not relevant. However, it is not trivial to do this without losing completeness (i.e. guarantee that all existing relevant plans can be found), because the fact that a plan is relevant not only depends on the state but also on the path. Then, determining whether or not there are more relevant plans remains a subject of future work. In this work we consider a time bound. But, if the top-k planning system could provide guarantees on loop-less paths completeness would be ensured.

Algorithm 1: IterativeRelevantTopK($\langle \Pi, k, \omega \rangle$)

```
$P_k \leftarrow \emptyset, \pi \leftarrow \emptyset$
**while** $|P_k| < k \wedge \neg timeout$ **do**
    $\pi \leftarrow get\_next\_plan(\Pi)$
    **if** $\pi = \emptyset$ **then**
        **return** $P_k$
    **end if**
    **if** $\omega(\pi)$ **then**
        $P_k \leftarrow P_k \cup \{\pi\}$
    **end if**
    $\Pi \leftarrow forbid(\Pi, \pi)$
**end while**
**return** $P_k$
```

Evaluation

Evaluation Setting

For this particular work, we wish to analyse the number of perfectly justified plans found when solving the top-k and top-k relevant planning problems. We perform this analysis over 300 planning tasks from 15 different domains that are widely used in plan and goal recognition research (Ramírez and Geffner 2009; Sohrabi, Riabov, and Udrea 2016; Pereira, Oren, and Meneguzzi 2020). We selected this benchmark[3] and not the standard International Planning Competition (IPC) benchmark for two main reasons. Firstly, IPC tasks are usually difficult to solve optimally, and thus computing large sets of plans using an iterative approach might be too time consuming. Secondly, these are domains and problems were used also in other settings as diverse planning (Roberts, Howe, and Ray 2014).

We modify Katz et al. (2018) software to find relevant plans in an iterative manner, following Algorithm 1. Since we are interested in perfectly justified plans, the condition to meet, ω, will be perfect justification for all the experiments reported. We use our implementation of the Π^{order} compilation to check if found plans are perfectly justified. To do this, we set it to solve the MLR (Minimal Length Reduction) problem. This implementation takes as input a planning task

[3]The set of planning tasks we used will be made publicly available.

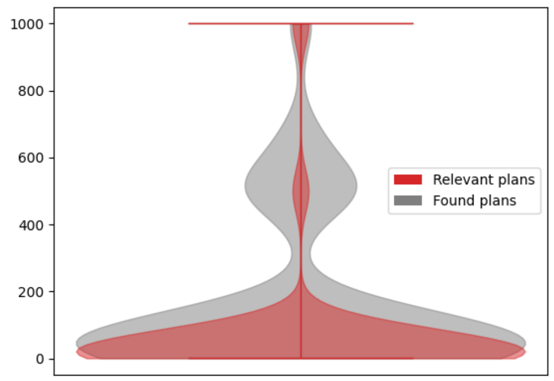

Figure 2: Violin plot of the distribution of the number of plans and relevant plans found in the kitchen domain.

(in PDDL) and a plan, and generates the Π^{order} task (also in PDDL). If the number of actions in the solution of this task is the same as in the original plan, then the plan did not have unnecessary actions. We use the Fast-Downward (FD) planning system (Helmert 2006) to solve the Π^{order} tasks, configured to find an optimal solution using A* as a search algorithm and the h_{max} heuristic (Bonet and Geffner 2001).

The experiments were run on Intel(R) Xeon(R) CPU X3470 @ 2.93GHz machines. We used a time limit of 15 minutes and a memory limit of 14GB to solve each top-k and top-k relevant problem. For each instance of each domain, we solve both the top-k and top-k relevant planning problems for $k \in [1, 5, 10, 50, 100, 500, 1000]$. We did not conduct experiments with larger values of k since most tasks run out of memory or time.

Results

Table 1 shows a summary of the results for the top-k relevant planning problem. For each domain and value of k we report two values: the mean number and standard deviation of plans found before reaching k or the time limit (column Plans), and the average number and standard deviation of those plans that where relevant (column R-Plans). Remember that in this particular setting we consider a plan to be relevant if it is perfectly justified.

When the number of R-Plans is smaller than k, this indicates that on average we were not able to find the desired amount of relevant plans. This happens often in some domains as blocks-world, campus, depots, grid and sokoban. In general, the ratio of Plans to R-Plans decreases as k increases. Currently we can not know if there are enough additional relevant plans to reach k. As explained previously, this is a subject of further research. But we can make some analysis considering specific domain characteristics. For instance, in blocks-world, many actions involving blocks that are not in the goals are not relevant and in general there are few relevant plans. This is reflected on the number of Plans vs. R-Plans for blocks-world.

	$k=5$		$k=10$		$k=50$		$k=100$		$k=500$		$k=1000$	
Domain	Plans	Rel-Plans	Plans	Rel-Plans	Plans	Rel-Plans	Plans	Rel-Plans	Plans	Rel-Plans	Plans	Rel-Plans
blocks	44.1 ± 65.09	4.95 ± 0.22	158.9 ± 125.41	8.55 ± 2.14	255.45 ± 82.27	22.2 ± 14.97	261.55 ± 69.82	22.9 ± 16.57	261.55 ± 69.82	22.9 ± 16.57	261.55 ± 69.82	22.9 ± 16.57
campus	8.2 ± 6.57	5.0 ± 0.0	13.2 ± 6.57	10.0 ± 0.0	120.5 ± 56.7	50.0 ± 0.0	261.15 ± 88.26	97.35 ± 8.16	445.9 ± 52.34	220.85 ± 72.27	445.9 ± 52.34	220.85 ± 72.27
depots	9.05 ± 18.11	5.0 ± 0.0	25.3 ± 52.57	9.9 ± 0.45	83.05 ± 71.18	44.5 ± 13.52	125.55 ± 53.73	87.0 ± 31.79	457.0 ± 94.55	417.25 ± 179.42	839.35 ± 296.94	797.25 ± 379.85
driverlog	5.0 ± 0.0	5.0 ± 0.0	10.9 ± 2.47	10.0 ± 0.0	100.65 ± 100.78	45.95 ± 9.78	149.95 ± 87.45	85.15 ± 29.1	434.2 ± 109.3	343.05 ± 201.86	696.25 ± 332.56	600.1 ± 434.46
dwr	5.0 ± 0.0	5.0 ± 0.0	10.0 ± 0.0	10.0 ± 0.0	67.6 ± 54.17	46.8 ± 9.85	112.6 ± 38.78	91.8 ± 25.24	440.35 ± 99.52	370.4 ± 182.27	721.05 ± 320.51	651.1 ± 409.12
ferry	5.0 ± 0.0	5.0 ± 0.0	10.0 ± 0.0	10.0 ± 0.0	55.0 ± 11.23	50.0 ± 0.0	117.8 ± 29.94	99.0 ± 3.64	420.6 ± 155.36	341.6 ± 165.52	565.8 ± 310.21	471.8 ± 342.68
grid	47.9 ± 63.38	4.55 ± 0.83	69.45 ± 64.73	7.7 ± 2.94	104.15 ± 53.6	14.45 ± 13.78	107.3 ± 51.12	15.95 ± 18.22	107.3 ± 51.12	15.95 ± 18.22	107.3 ± 51.12	15.95 ± 18.22
intruder	5.0 ± 0.0	5.0 ± 0.0	10.0 ± 0.0	10.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	500.0 ± 0.0	500.0 ± 0.0	1000.0 ± 0.0	1000.0 ± 0.0
kitchen	5.0 ± 0.0	5.0 ± 0.0	10.0 ± 0.0	10.0 ± 0.0	406.15 ± 269.05	38.8 ± 8.79	423.65 ± 244.68	56.3 ± 32.99	563.65 ± 52.84	196.3 ± 228.66	738.65 ± 198.02	371.3 ± 473.33
logistics	5.0 ± 0.0	5.0 ± 0.0	10.0 ± 0.0	10.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	500.0 ± 0.0	500.0 ± 0.0	1000.0 ± 0.0	1000.0 ± 0.0
miconic	5.0 ± 0.0	5.0 ± 0.0	10.0 ± 0.0	10.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	395.1 ± 139.83	383.2 ± 124.99	558.5 ± 370.72	518.8 ± 308.3
rover	5.0 ± 0.0	5.0 ± 0.0	10.0 ± 0.0	10.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	112.25 ± 54.78	99.7 ± 1.34	499.05 ± 47.3	475.2 ± 91.95	914.15 ± 174.57	859.6 ± 266.66
satellite	5.4 ± 0.82	5.0 ± 0.0	11.2 ± 4.02	10.0 ± 0.0	63.5 ± 35.14	50.0 ± 0.0	127.0 ± 59.89	99.3 ± 3.13	524.05 ± 169.89	327.95 ± 115.18	558.85 ± 202.5	361.55 ± 197.39
sokoban	17.8 ± 20.11	5.0 ± 0.0	47.95 ± 43.0	9.45 ± 1.23	92.55 ± 45.05	23.4 ± 15.3	95.45 ± 42.66	26.3 ± 21.15	95.45 ± 42.66	26.3 ± 21.15	95.45 ± 42.66	26.3 ± 21.15
zenotravel	5.0 ± 0.0	5.0 ± 0.0	10.0 ± 0.0	10.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	109.3 ± 28.77	99.0 ± 4.47	444.85 ± 130.84	396.55 ± 132.22	716.85 ± 477.12	602.75 ± 342.55

Table 1: Results for top-k relevant plans. ω is perfect justification. Plans column shows the mean number of plans found. R-Plans shows the mean number of relevant plans found.

	$k=5$		$k=10$		$k=50$		$k=100$		$k=500$		$k=1000$	
Domain	TOP-K	TOP-K-R	TOP-K	TOP-K-R	TOP-K	TOP-K-R	TOP-K	TOP-K-R	TOP-K	TOP-K-R	TOP-K	TOP-K-R
blocks	0.61 ± 0.21	61.79 ± 197.31	1.28 ± 0.45	403.32 ± 407.96	13.01 ± 6.0	848.78 ± 194.22	71.33 ± 47.74	892.13 ± 2.83	892.67 ± 4.58	892.13 ± 2.83	892.67 ± 4.58	892.13 ± 2.83
campus	0.6 ± 0.1	3.42 ± 3.01	1.06 ± 0.17	5.32 ± 2.82	7.03 ± 3.05	73.14 ± 47.69	23.96 ± 7.16	305.19 ± 248.23	856.96 ± 77.69	885.98 ± 2.37	893.99 ± 1.86	885.98 ± 2.37
depots	4.46 ± 4.77	11.42 ± 24.09	4.54 ± 4.7	57.11 ± 197.6	7.04 ± 6.08	156.38 ± 318.39	19.93 ± 29.91	167.9 ± 313.52	183.9 ± 363.04	290.95 ± 307.78	227.08 ± 393.18	429.71 ± 274.82
driverlog	0.32 ± 0.22	1.46 ± 0.28	0.43 ± 0.29	3.14 ± 1.58	3.96 ± 7.23	188.14 ± 356.62	17.23 ± 35.84	239.66 ± 381.36	352.7 ± 431.0	453.29 ± 398.78	446.25 ± 457.37	543.48 ± 348.29
dwr	1.65 ± 1.74	4.46 ± 3.56	1.66 ± 1.74	6.83 ± 5.06	2.7 ± 3.2	113.46 ± 265.99	10.0 ± 25.22	135.88 ± 259.73	358.41 ± 447.01	547.2 ± 304.83	358.45 ± 446.98	718.61 ± 237.36
ferry	0.25 ± 0.04	1.6 ± 0.18	0.26 ± 0.04	2.72 ± 0.31	4.11 ± 9.19	26.6 ± 45.9	39.61 ± 79.46	165.11 ± 296.76	499.92 ± 399.58	609.16 ± 344.1	691.51 ± 355.47	738.12 ± 247.45
grid	2.17 ± 2.98	277.25 ± 412.79	7.07 ± 11.28	459.18 ± 438.24	203.9 ± 247.39	815.92 ± 200.78	609.08 ± 324.62	880.73 ± 23.99	883.33 ± 11.98	880.73 ± 23.99	883.33 ± 11.98	880.73 ± 23.99
intruder	0.2 ± 0.02	1.47 ± 0.06	0.19 ± 0.02	2.52 ± 0.11	0.2 ± 0.02	10.92 ± 0.48	0.21 ± 0.02	21.44 ± 0.96	0.25 ± 0.01	105.21 ± 4.82	0.28 ± 0.01	209.76 ± 9.63
kitchen	0.22 ± 0.04	1.44 ± 0.25	0.31 ± 0.12	2.78 ± 0.36	1.47 ± 1.02	579.35 ± 426.1	5.01 ± 3.58	583.86 ± 419.8	291.94 ± 228.38	619.15 ± 370.45	579.58 ± 436.1	663.62 ± 308.28
logistics	0.22 ± 0.02	1.55 ± 0.04	0.21 ± 0.02	2.63 ± 0.06	0.22 ± 0.02	11.36 ± 0.24	0.23 ± 0.02	22.3 ± 0.47	0.29 ± 0.03	109.28 ± 2.28	0.33 ± 0.03	218.01 ± 4.63
miconic	0.27 ± 0.06	1.47 ± 0.1	0.42 ± 0.23	2.85 ± 0.49	4.32 ± 4.62	16.91 ± 6.6	23.14 ± 28.47	48.37 ± 32.12	579.04 ± 391.23	642.95 ± 330.82	700.86 ± 334.11	771.58 ± 228.86
rover	0.42 ± 0.48	1.74 ± 0.53	0.42 ± 0.48	2.86 ± 0.57	0.43 ± 0.48	11.76 ± 1.05	0.58 ± 0.81	66.12 ± 192.94	48.66 ± 199.57	190.02 ± 236.65	223.99 ± 397.08	391.2 ± 288.52
satellite	0.28 ± 0.14	1.47 ± 0.31	0.38 ± 0.21	2.83 ± 1.49	3.18 ± 2.55	22.86 ± 30.06	12.89 ± 16.98	84.58 ± 188.72	571.24 ± 328.02	817.98 ± 204.97	846.09 ± 198.99	848.75 ± 153.6
sokoban	6.99 ± 10.57	55.75 ± 115.06	19.87 ± 29.48	297.85 ± 374.79	310.81 ± 312.7	811.89 ± 177.28	635.6 ± 297.76	883.9 ± 10.72	882.85 ± 13.61	883.9 ± 10.72	882.85 ± 13.61	883.9 ± 10.72
zenotravel	0.61 ± 0.58	1.76 ± 0.67	0.73 ± 0.78	2.86 ± 1.02	4.2 ± 6.19	15.09 ± 7.91	15.18 ± 24.28	80.94 ± 192.4	379.63 ± 384.06	505.01 ± 378.75	614.63 ± 412.74	687.66 ± 308.04

Table 2: Time results for the top-k and top-k relevant problems.

	$k=5$		$k=10$		$k=50$		$k=100$		$k=500$		$k=1000$	
Domain	K	K-R	K	K-R	K	K-R	K	K-R	K	K-R	K	K-R
blocks	0	1	0	7	0	19	0	20	20	20	20	20
campus	0	0	0	0	0	0	0	2	13	20	20	20
depots	0	0	0	1	0	3	0	3	4	4	5	5
driverlog	0	0	0	0	0	4	0	5	7	9	10	10
dwr	0	0	0	0	0	2	0	2	8	8	8	12
ferry	0	0	0	0	0	0	0	2	9	12	15	16
grid	0	6	0	10	2	18	10	20	20	20	20	20
intruder	0	0	0	0	0	0	0	0	0	0	0	0
kitchen	0	0	0	0	0	13	0	13	0	13	13	13
logistics	0	0	0	0	0	0	0	0	0	0	0	0
miconic	0	0	0	0	0	0	0	0	11	12	15	16
rover	0	0	0	0	0	0	0	1	1	2	5	5
satellite	0	0	0	0	0	0	0	1	9	18	19	19
sokoban	0	0	0	5	3	17	10	20	20	20	20	20
zenotravel	0	0	0	0	0	0	0	1	6	9	13	14

Table 3: Number of timeouts recorded for the top-k and top-k relevant planning problems

There are several domains for which a really large amount of plans must be found to find a small amount of relevant plans, and in most cases the desired number k is never achieved. In contrast, there are domains like logistics where all found plans are relevant. This occurs because a single plan for logistics can be used to derive additional plans by reducing it to a partial order plan. Each plan derived from the first (optimal) plan is perfectly justified because this is an unit-cost domain. Even more, in all our instances the all the desired (k) plans are derived from the first plan found. Figure 2 shows the distribution of the number of plans and relevant plans in the kitchen domain. This is one of the domains where the number of non-relevant plans is especially high. The width of the shadows shows that more data points share that value. The upper and lower horizontal lines represent the maximum an minimum values of the results respectively. The vertical lines correspond to the medians. The median of the first is 50 while for the second is 10. It is straightforward to verify that the presence of relevant plans considerably decreases as larger k values are required. As expected, irrelevant information in plans grows as more iterations are carried out.

Table 2 shows the required time (in seconds) to solve the top-k (TOP-K) and top-k relevant (TOP-K-R) problems for different values of k. As expected, the time spent on verifying that the found plans are relevant is high, given that guaranteeing a plan does not contain redundant actions is NP-complete (Fink and Yang 1992; Nakhost and Müller 2010). There are domains, as blocks-world, grid and kitchen, where most of the found plans are not relevant, and therefore this time can be better justified than for other domains, like logistic, where all found plans are relevant. This can motivate further investigation on how to determine if the extra time needed to solve the top-k relevant problem is worth it depending on domain/problem characteristics.

Table 3 shows the number of timeouts for each domain and for top-k (K) and top-k relevant (K-R). As expected, the number of timeouts for the relevant variant greatly exceeds the one for the regular top-k. Note that with the current approach if there are not k relevant plans our approach always stops due to the time limit.

Related Work

There exists several approaches to top-k planning (Riabov, Sohrabi, and Udrea 2014; Katz et al. 2018; Speck, Mattmüller, and Nebel 2020), but so far all of them are concerned with the problem of finding the best k plans for a planning task, and not much has been studied the semantics and utility of these plans. Closer works to the idea introduced in this paper are those about diverse planning (Srivastava et al. 2007; Roberts, Howe, and Ray 2014; Katz and Sohrabi 2020), where resulting plans are required to be different and similarity metrics are defined between plans. We consider the notion of plan relevance that applies only to single plans and can be considered as complementary to plan diversity. In applications of top-k planning one might want to generate plans that are at the same time diverse and relevant.

In this work, the condition for plan relevance is based on perfectly justified plans. Specifically we filter those plans that are not perfectly justified in a post-optimization step. There are some other works on plan post-optimization. Fink and Yang [1992] formalized different notions of plan justifications and provided complexity results for them. Specifically, they defined greedily justified actions as those that make the plan invalid when they are removed from it, and perfectly-justified plans as those with no redundant actions. Nakhost and Müller [2010] proposed Action Elimination, an algorithm based on greedy justification, and an additional technique based on plan neighborhood graph search. There are methods based on identifying redundant actions and non-optimal sub-plans by analyzing action dependencies, independencies (Chrpa, McCluskey, and Osborne 2012b), by checking pairs of inverse actions (Chrpa, McCluskey, and Osborne 2012a), and SAT-based approaches (Balyo, Chrpa, and Kilani 2014; Muise, Beck, and McIlraith 2016). The work in this paper is closely related to all these works with the difference that we approach the problem using Automated Planning. However, our method could be replaced by any other.[4]

There are also techniques that remove irrelevant information at preprocessing. For instance, Nebel, Dimopoulos, and Koehler 1997 proposed heuristics for selecting relevant information based on minimizing the number of initial facts by computing a fact generation tree going backwards from the goals; and a recent approach (Silver et al. 2020) learns convolutional graph neural networks to predict subsets of objects that are sufficient for solving the planning task. Approaching the problem at preprocessing has the additional advantage that it can make easier the planning process. This is specially interesting when the number of objects is very large. In this case, most modern heuristic planners that ground the actions over objects during preprocessing scale poorly. This is also one of the motivations for recent research on lifted planning (Corrêa et al. 2020), abstractions that simplify the problem (Fuentetaja and de la Rosa 2016) and some approaches based on generalized planning, as the aforementioned work of Silver et al.. We believe that studying techniques that can be applied in preprocessing or even during search in the context of top-k relevant planning would be an interesting research direction.

[4]We have some results comparing it to the Max-SAT approach of Balyo, Chrpa, and Kilani (2014), showing both very similar performance times.

Conclusions and Future Work

In this work we proposed the idea of top-k relevant plans as plans that meet some extra condition in the context of top-k planning. Specifically, we consider plans that are perfectly justified (i.e. they do not contain subsets of actions that can be removed while maintaining plan validity). We have incorporated this notion into a top-k planner as a filtering step which is applied every time a new plan is found. The filtering process is posed as an Automated Planning task. In particular, given a plan we show how to create planning tasks to solve Minimal Length Reduction (MLR) problem. We have performed experiments in a variety of domains. Many problems have few relevant plans or the number of plans found in order to find the desired relevant plans is high.

Regarding the proposed approach, determining whether there exist additional relevant plans to those already found is an interesting line of future work. We also plan to consider our work in conjunction to additional techniques that can be applied to filter irrelevant information in a preprocessing step or during search. Additionally, we wish to consider different definitions of relevance. Finally, we want to study the impact of plan relevance in applications of top-k planning as goal recognition.

Acknowledgments

This work has been partially funded by FEDER/Ministerio de Ciencia, Innovación y Universidades - Agencia Estatal de Investigación/TIN2017-88476-C2-2-R, RTC-2016-5407-4, and the Madrid Government (Comunidad de Madrid-Spain) under the Multiannual Agreement with UC3M in the line of Excellence of University Professors (EPUC3M17), and in the context of the V PRICIT (Regional Programme of Research and Technological Innovation)."

References

Balyo, T.; Chrpa, L.; and Kilani, A. 2014. On different strategies for eliminating redundant actions from plans. In *Seventh Annual Symposium on Combinatorial Search*.

Boddy, M. S.; Gohde, J.; Haigh, T.; and Harp, S. A. 2005. Course of Action Generation for Cyber Security Using Classical Planning. In *ICAPS 2005*, 12–21.

Bonet, B.; and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence*, 129(1): 5–33.

Borrajo, D.; Gopalakrishnan, S.; and Potluru, V. K. 2020. Goal Recognition via Model-based and Model-free Techniques. In *ICAPS Workshop on Planning for Financial Services (FinPlan)*.

Borrajo, D.; and Veloso, M. 2020. Domain-independent Generation and Classification of Behavior Traces. In *ICAPS Workshop on Planning for Financial Services (FinPlan)*.

Borrajo, D.; Veloso, M.; and Shah, S. 2020. Simulating and Classifying Behavior in Adversarial Environments Based on Action-State Traces: An Application to Money Laundering. In *Proceedings of the 2020 ACM International Conference on AI in Finance*. New York (EEUU).

Celorrio, S. J.; Haslum, P.; Thiebaux, S.; et al. 2013. Pruning bad quality causal links in sequential satisfying planning.

Chrpa, L.; McCluskey, T. L.; and Osborne, H. 2012a. Determining redundant actions in sequential plans. In *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, volume 1, 484–491. IEEE.

Chrpa, L.; McCluskey, T. L.; and Osborne, H. 2012b. Optimizing plans through analysis of action dependencies and independencies. In *Twenty-Second International Conference on Automated Planning and Scheduling*.

Corrêa, A. B.; Pommerening, F.; Helmert, M.; and Frances, G. 2020. Lifted successor generation using query optimization techniques. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, 80–89.

Fikes, R. E.; and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4): 189–208.

Fink, E.; and Yang, Q. 1992. Formalizing plan justifications.

Fuentetaja, R.; and de la Rosa, T. 2016. Compiling irrelevant objects to counters. special case of creation planning. *AI Communications*, 29(3): 435–467.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26: 191–246.

Katz, M.; and Sohrabi, S. 2020. Reshaping diverse planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 9892–9899.

Katz, M.; Sohrabi, S.; and Udrea, O. 2020. Top-Quality Planning: Finding Practically Useful Sets of Best Plans. In *AAAI 2020*, 9900–9907.

Katz, M.; Sohrabi, S.; Udrea, O.; and Winterer, D. 2018. A Novel Iterative Approach to Top-k Planning. In *ICAPS 2018*, 132–140.

Muise, C.; Beck, J. C.; and McIlraith, S. A. 2016. Optimal partial-order plan relaxation via MaxSAT. *Journal of Artificial Intelligence Research*, 57: 113–149.

Mund, S.; Vallati, M.; and McCluskey, T. L. 2020. An Exploration of the Use of AI Planning for Predicting Stock Market Movement. In *ICAPS 2020 Workshop on AI Planning for Financial Services (FinPlan)*.

Nakhost, H.; and Müller, M. 2010. Action Elimination and Plan Neighborhood Graph Search: Two Algorithms for Plan Improvement. In Brafman, R. I.; Geffner, H.; Hoffmann, J.; and Kautz, H. A., eds., *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010, Toronto, Ontario, Canada, May 12-16, 2010*, 121–128. AAAI.

Nebel, B.; Dimopoulos, Y.; and Koehler, J. 1997. Ignoring irrelevant facts and operators in plan generation. In *European Conference on Planning*, 338–350. Springer.

Pereira, R. F.; Oren, N.; and Meneguzzi, F. 2020. Landmark-based approaches for goal recognition as planning. *Artificial Intelligence*, 279: 103217.

Pozanco, A.; Polychroniadou, A.; Magazzeni, D.; and Borrajo, D. 2021. Proving Security of Cryptographic Protocols using Automated Planning. In *ICAPS Workshop on Planning for Financial Services (FinPlan)*.

Ramírez, M.; and Geffner, H. 2009. Plan recognition as planning. In *Twenty-First International Joint Conference on Artificial Intelligence*.

Ramírez, M.; and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Riabov, A.; Sohrabi, S.; and Udrea, O. 2014. New algorithms for the top-k planning problem. In *Proceedings of the scheduling and planning applications workshop (spark) at the 24th international conference on automated planning and scheduling (icaps)*, 10–16.

Roberts, M.; Howe, A. E.; and Ray, I. 2014. Evaluating diversity in classical planning. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*.

Silver, T.; Chitnis, R.; Curtis, A.; Tenenbaum, J.; Lozano-Perez, T.; and Kaelbling, L. P. 2020. Planning with learned object importance in large problem instances using graph neural networks. *arXiv preprint arXiv:2009.05613*.

Sohrabi, S.; Riabov, A.; and Udrea, O. 2016. Plan Recognition as Planning Revisited. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*.

Speck, D.; Mattmüller, R.; and Nebel, B. 2020. Symbolic Top-k Planning. In *AAAI 2020*, 9967–9974.

Srivastava, B.; Nguyen, T. A.; Gerevini, A.; Kambhampati, S.; Do, M. B.; and Serina, I. 2007. Domain Independent Approaches for Finding Diverse Plans. In *IJCAI*, 2016–2022.