Top-Quality: Finding Practically Useful Sets of Best Plans

Submission number #

Abstract

The need for finding a set of plans rather than one has been motivated by a variety of planning applications. The problem is studied in the context of both diverse and top-k planning: while diverse planning focuses on the difference between pairs of plans, the focus of top-k planning is on the quality of each individual plan. Recent work in diverse planning introduced additionally restrictions on solution quality. Naturally, there are application domains where diversity plays the major role and domains where quality is the predominant feature. In both cases, however, the amount of produced plans is somewhat an artificial constraint, and the actual number has little meaning.

Inspired by the recent work in diverse planning, we propose a new computational problem called *top-quality* planning, where solution validity is defined through plan quality bound rather than an arbitrary number of plans. Switching to bounding plan quality allows us to implicitly represent sets of plans. In particular, it makes it possible to represent sets of valid plan reorderings with one plan. We formally define the corresponding computational problem and present the first planner for that problem. We empirically demonstrate the superior performance of our approach compared to a top-k planner-based baseline, ranging from 49% increase in coverage for finding all optimal plans to 69% increase in coverage for finding all plans of quality up to 120% of optimal plan cost.

1 Introduction

While the main focus in classical planning was on producing a single plan, a variety of applications has shown the need for finding a set of plans rather than one. These applications include malware detection (Boddy *et al.* 2005), plan recognition as planning and its applications (Riabov *et al.* 2015; Sohrabi *et al.* 2016a; Sohrabi *et al.* 2018; Shvo *et al.* 2018), human team planning (Kim *et al.* 2018), explainable AI (Chakraborti *et al.* 2018), re-planning and plan monitoring (Fox *et al.* 2006).

The problem of finding a set of plans is studied in the context of both diverse planning (e.g., (Nguyen *et al.* 2012)) and top-k planning (e.g., (Katz *et al.* 2018)). Diverse planning focuses on the difference between pairs of plans, evaluating a set of plans by aggregating over the pairwise differences between plans in the set. Recent work in diverse planning introduced additional restrictions on solution quality, requiring each plan in the set to also be of bounded quality (Vadlamudi and Kambhampati 2016). Top-k planning is a generalization of cost-optimal planning. The focus of top-k planning is on the quality of each individual plan, guaranteeing that no plan of better cost exists outside the solution of a requested size.

Naturally, there are application domains where diversity plays the major role and domains where quality is the predominant feature. The latter include plan recognition (Sohrabi *et al.* 2016a), multi-agent plan recognition (Shvo *et al.* 2018), human team planning (Kim *et al.* 2018), and explainable AI (Chakraborti *et al.* 2018). These applications exploit top-k planners to derive a large number of plans. In these domains, though, the focus on the number of plans provided is somewhat artificial, and is intended solely to ensure that a sufficient number of plans is found. Further, ordering of actions in a plan can be of less importance in some applications. Plan recognition is one such example application. In plan recognition as planning (Sohrabi *et al.* 2016a; Shvo *et al.* 2018), a planning tasks consists of actions that explain/discard observations. There is no meaning to the order among these actions. Some specific practical applications for plan recognition are hypothesis generation (Sohrabi *et al.* 2016b) and scenario planning advisor (Sohrabi *et al.* 2018). These applications use a top-k planner with a large bound on the number of required plans k, and the obtained plans are post-processed to discard reorderings and cluster similar plans. This would also apply to e.g., problems with actions that correspond to information gathering, where no particular ordering is required. The clear disadvantage of a top-k planner in such cases is that it would generate all possible orderings before proceeding to plans of a higher cost. Thus, the number of required plans used in practice is often a crude over-approximation. Further, even quite large numbers are often not sufficient to ensure that the set of plans includes enough plans of interest, since plans can easily have millions of valid reorderings. A top-k planner would have to generate all these plans before it can get to a plan of a higher cost. Diverse planners (Bryce 2014; Nguyen *et al.* 2012; Coman and Muñoz-Avila 2011; Roberts *et al.* 2014; Vadlamudi and Kambhampati 2016) tackle the issue by defining diversity criteria over a set of plans, but only a handful of

Copyright © 2019, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

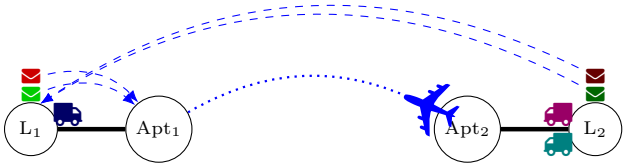

Figure 1: Example logistics task.

work take the plan quality under consideration (Roberts *et al.* 2014; Vadlamudi and Kambhampati 2016). However, these diverse planners still do not provide the guarantees on solution quality that top-k planners provide.

In this paper, we propose a new computational problem called *top-quality planning*. The objective of top-quality planning is to find a set of all plans of bounded quality, for a given (absolute) bound. That is, we suggest an alternative definition of solution validity, by bounding the solution quality instead of bounding the number of plans. This allows us to implicitly represent sets of plans without knowing the exact number of plans in the set. In particular, in this work, we represent sets of all possible reorderings of each plan by one representative plan. Furthermore, we propose a first planner for top-quality planning that iteratively finds a single plan of top quality and forbids at once all plans found so far, including all their possible reorderings. For that, we adapt a recently proposed diverse planning reformulation that forbids a single multiset of actions (Anonymous 2019) to forbid exactly a collection of multisets. Our adaptation of the existing reformulation allows us to forbid multiple sets of plans at each iteration while preserving soundness and completeness of our approach. We empirically demonstrate the superior performance of our approach compared to a top-k planner with a large k bound, ranging from 49% increase in coverage for finding all optimal plans to 69% increase in coverage for finding all plans of quality up to 120% of optimal plan cost.

2 Preliminaries

We consider classical planning tasks in the well-known SAS$^+$ formalism (Bäckström and Nebel 1995), extended with action costs. Such *planning tasks* $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$ consist of $\mathcal{V}$, a finite set of finite-domain *state variables*, $\mathcal{O}$, a finite set of *actions*, s_0, an *initial state*, and $s_\star$, the *goal*. Each variable $v \in \mathcal{V}$ is associated with a finite domain $\mathcal{D}(v)$ of variable values. These variable, value pairs are called *facts*. A *partial assignment* p maps a subset of variables $vars(p) \subseteq \mathcal{V}$ to values in their domains. For a variable $v \in \mathcal{V}$ and partial assignment p, the value of v in p is denoted by $p[v]$ if $v \in vars(p)$ and we say $p[v]$ is *undefined* if $v \notin vars(p)$. A partial assignment s with $vars(s) = \mathcal{V}$, is called a *state*. State s is *consistent* with partial assignment p if they agree on all variables in $vars(p)$, shortly denoted by $p \subseteq s$. The product $\mathcal{S} = \prod_{v \in \mathcal{V}} \mathcal{D}(v)$ is called the *state space* of planning task Π. s_0 is a state and $s_\star$ is a partial assignment. A state s is called a *goal state* if $s_\star \subseteq s$ and the set of all goal states is denoted by $\mathcal{S}_{s_\star}$. Each action o in $\mathcal{O}$ is a pair $\langle pre(o), \mathit{eff}(o) \rangle$ where $pre(o)$ is a partial assignment called *precondition* and $\mathit{eff}(o)$ is a partial assignment called *effect*. Further, o has an associated natural number $cost(o)$, called *cost*. An action o is applicable in state s if $pre(o) \subseteq s$. Applying action o in state s results in a state denoted by $s[\![o]\!]$ where $s[\![o]\!][v] = \mathit{eff}(o)[v]$ for all $v \in vars(\mathit{eff})$ and $= s[\![o]\!][v] = s[v]$ for all other variables. An action sequence $\pi = \langle o_1, \cdots, o_n \rangle$ is applicable in state s if there are states $s_0, \cdots, s_n$ such that o_i is applicable in s_{i-1} and $s_{i-1}[\![o_i]\!] = s_i$ for $0 \leq i \leq n$. We denote s_n by $s[\![\pi]\!]$. For convenience we often write $o_1, \cdots, o_n$ instead of $\langle o_1, \cdots, o_n \rangle$. An action sequence with $s[\![\pi]\!] \in \mathcal{S}_{s_\star}$ is called a *plan*. The cost of a plan π, denoted by $cost(\pi)$ is the summed cost of the actions in the plan. For a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$, the set of all plans is denoted by $\mathcal{P}_\Pi$. A plan π is *optimal* if its cost is minimal among all plans in $\mathcal{P}_\Pi$. For a plan π, we denote by $\mathrm{MS}(\pi)$ the multiset[1] of actions in π. Note that two different plans π and π' can have $\mathrm{MS}(\pi) = \mathrm{MS}(\pi')$. We call such plans *reordering* of each other. Reoderings of actions of a plan that are plans are called *valid* reorderings.

π_a	π_b	π_c
(load P_4 T_2 L_2)	(load P_3 T_2 L_2)	(load P_4 T_3 L_2)
(load P_3 T_2 L_2)	(load P_4 T_2 L_2)	(load P_3 T_3 L_2)
(drive T_2 L_2 Apt_2)	(drive T_2 L_2 Apt_2)	(drive T_3 L_2 Apt_2)
(unload P_4 T_2 Apt_2)	(unload P_4 T_2 Apt_2)	(unload P_4 T_3 Apt_2)
(unload P_3 T_2 Apt_2)	(unload P_3 T_2 Apt_2)	(unload P_3 T_3 Apt_2)
(load P_2 T_1 L_1)	(load P_2 T_1 L_1)	(load P_2 T_1 L_1)
(load P_1 T_1 L_1)	(load P_1 T_1 L_1)	(load P_1 T_1 L_1)
(load P_3 A Apt_2)	(load P_3 A Apt_2)	(load P_3 A Apt_2)
(load P_4 A Apt_2)	(load P_4 A Apt_2)	(load P_4 A Apt_2)
(fly A Apt_2 Apt_1)	(fly A Apt_2 Apt_1)	(fly A Apt_2 Apt_1)
(unload P_3 A Apt_1)	(unload P_3 A Apt_1)	(unload P_3 A Apt_1)
(unload P_4 A Apt_1)	(unload P_4 A Apt_1)	(unload P_4 A Apt_1)
(drive T_1 L_1 Apt_1)	(drive T_1 L_1 Apt_1)	(drive T_1 L_1 Apt_1)
(load P_4 T_1 Apt_1)	(load P_4 T_1 Apt_1)	(load P_4 T_1 Apt_1)
(load P_3 T_1 Apt_1)	(load P_3 T_1 Apt_1)	(load P_3 T_1 Apt_1)
(unload P_2 T_1 Apt_1)	(unload P_2 T_1 Apt_1)	(unload P_2 T_1 Apt_1)
(unload P_1 T_1 Apt_1)	(unload P_1 T_1 Apt_1)	(unload P_1 T_1 Apt_1)
(drive T_1 Apt_1 L_1)	(drive T_1 Apt_1 L_1)	(drive T_1 Apt_1 L_1)
(unload P_4 T_1 L_1)	(unload P_4 T_1 L_1)	(unload P_4 T_1 L_1)
(unload P_3 T_1 L_1)	(unload P_3 T_1 L_1)	(unload P_3 T_1 L_1)

Figure 2: Three example cost-optimal plans for the example task.

In this paper, we will use a logistics task as our running example. This task has two cities, with two locations each, L_1 and L_2, three trucks, T_1, T_2, T_3, that can drive within their cities, one airplane, A, that can fly between the airport locations, Apt_1 and Apt_2, and four packages, P_1 to P_4, that need to be transported from their initial locations to some specified goal locations. Figure 1 shows the initial and goal locations of all objects. Assuming all actions having unit cost, a cost-optimal plan for this task will consists of 20 actions. Example plans π_a, π_b, and π_c are depicted in Figure 2.

Given a plan π, it is sometimes possible to obtain a different plan of equivalent cost without solving the planning task again. Two of such possible ways: action reordering and de-

[1] A set with possible multiple occurences of the same element.

riving symmetric plans, are exploited by state-of-the-art top-k planners (Katz *et al.* 2018). While action reordering is performed using search and may be time consuming, symmetric plans can be obtained using structural symmetries (Shleyfman *et al.* 2015). Structural symmetries are permutations of variable values and actions that induce automorphisms of the state transition graph. Here, we present the definition of structural symmetries for SAS^+ as was given by Sievers *et al.* (2017).

Definition 1 (structural symmetry) *For a* SAS^+ *planning task* $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$*, let* F *be the set of* Π*'s facts, i. e. pairs* $\langle v, d \rangle$ *with* $v \in \mathcal{V}$*,* $d \in dom(v)$*. A* structural symmetry *for* Π *is a permutation* $\sigma : \mathcal{V} \cup F \cup \mathcal{O} \to \mathcal{V} \cup F \cup \mathcal{O}$*, where:*

1. $\sigma(\mathcal{V}) = \mathcal{V}$ *and* $\sigma(F) = F$ *such that* $\sigma(\langle v, d \rangle) = \langle v', d' \rangle$ *implies* $v' = \sigma(v)$*;*
2. $\sigma(\mathcal{O}) = \mathcal{O}$ *such that for* $o \in \mathcal{O}$*,* $\sigma(pre(o)) = pre(\sigma(o))$*,* $\sigma(eff(o)) = eff(\sigma(o))$*,* $cost(\sigma(o)) = cost(o)$*;*
3. $\sigma(s_\star) = s_\star$*;*

where $\sigma(\{x_1, \dots, x_n\}) := \{\sigma(x_1), \dots, \sigma(x_n)\}$*, and for a partial state* s*,* $s' := \sigma(s)$ *is the partial state obtained from* s *s.t. for all* $v \in \mathcal{V}(s)$*,* $\sigma(\langle v, s[v] \rangle) = \langle v', d' \rangle$ *implies* $s'[v'] = d'$*.*

Given a plan $\pi = o_1 \dots o_n$ and a structural symmetry σ that stabilizes the initial state, applying the permutation σ to each action in the plan results in a necessarily valid plan $\sigma(\pi) = \sigma(o_1) \dots \sigma(o_n)$ of the same cost. Note that $\sigma(\pi)$ is not a reordering of π, since σ may map actions from π to actions outside of π.

In our example, the structural symmetries can detect symmetries between two of the trucks T_2 and T_3, between the two packages that are initially in L_1, and between the two packages that are initially in L_2. Thus, structural symmetries can be used to obtain additional plans from a given plan. In our example, the plan π_c in Figure 2 can be obtained from π_a using the symmetry between the trucks T_2 and T_3. Note that these two plans use different actions and thus are not reorderings of each other. The plan π_b, on the other hand, is a reordering of π_a, changing the order between the first two actions. These two plans are not symmetric, since mapping the action (load P_4 T_2 L_2) to (load P_3 T_2 L_2) would also require mapping (unload P_4 T_2 L_2) to (unload P_3 T_2 L_2). Naturally, there exist pairs of plans that are both symmetric and reordering of each other. There are 6602112 cost-optimal plans in our example, half of them are reorderings of the plan π_a and the other half are reordering of π_c.

Lastly, the *top-k planning problem* (Sohrabi *et al.* 2016b; Katz *et al.* 2018) is defined as follows.

Definition 2 (top-k planning problem) *Given a planning task* $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$ *and a natural number* k*, find a set of plans* $P \subseteq \mathcal{P}_\Pi$ *such that:*

(i) for all plans $\pi \in P$*, if there exists a plan* π' *for* Π *with* $cost(\pi') < cost(\pi)$*, then* $\pi' \in P$*, and*

(ii) $|P| \leq k$*, with* $|P| < k$ *implying* $P = \mathcal{P}_\Pi$*.*

An instance of the top-k planning problem $\langle \Pi, k \rangle$*, is called solvable if* $|P| = k$ *and unsolvable if* $|P| < k$*.*

The objective of top-k planning is finding k plans of lowest costs for a planning task Π and thus optimal planning is the special case of top-1 planning.

3 Top-quality Planning

We start by formally defining the top-quality planning problem as the problem of finding all plans of bounded quality.

Definition 3 (top-quality planning problem) *Given a planning task* $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$ *and a natural number* q*, find the set of plans* $P = \{\pi \in \mathcal{P}_\Pi \mid cost(\pi) \leq q\}$*.*

The top-quality planning problem is well-defined and always has a solution. Note that one can exploit existing tools for top-k planning to derive solutions to the top-quality planning problem, by setting k to a large value and adding another stopping criteria, once a plan π of $cost(\pi) > q$ was obtained. In such cases, P would explicitly contain all plans with $cost(\pi) \leq q$. This was done by Vadlamudi and Kambhampati (2016) as the first step in their algorithm, although they do not formally define the top-quality problem. These explicit sets of plans can get prohibitively large. Further, some of the plans in that set, although different as sequences of actions, could be considered equivalent from the underlying application perspective. If, in addition, it would be possible to escape the need for generating all these equivalent plans, the performance of the planners could improve significantly.

Let N be some equivalence relation on the set of plans $\mathcal{P}_\Pi$. For a plan $\pi \in \mathcal{P}_\Pi$, by $N[\pi]$ we denote the equivalence class of π, which is a set of all plans that are equivalent to π under N. Using that equivalence relation, we can define the quotient top-quality problem as follows.

Definition 4 (quotient top-quality planning problem) *Given a planning task* $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$*, an equivalence relation* N *over its set of plans* $\mathcal{P}_\Pi$*, and a natural number* q*, find a set of plans* $P \subseteq \mathcal{P}_\Pi$ *such that* $\bigcup_{\pi \in P} N[\pi]$ *is the solution to the top-quality planning problem.*

For equivalence relations that preserve plan cost the quotient top-quality planning problem always has a solution. Note that solutions to top-quality planning are solutions to the quotient top-quality planning under the identity equivalence relation. Further, while there is one possible solution to the top-quality planning problem, there can be many solutions to a quotient top-quality problem, defined by representatives of each equivalence class. Further, nothing in our definition prevents a solution from including more than one plan per equivalence class, the only restriction is that all equivalence classes have to be represented.

In this paper, we focus on one specific equivalence relation, considering two plans to be equivalent if their action multisets are. Formally, we consider the equivalence relation

$$U(\Pi) = \{(\pi, \pi') \mid \pi, \pi' \in \mathcal{P}_\Pi, \text{MS}(\pi) = \text{MS}(\pi')\}.$$

Thus, the main computational problem we consider in this paper is as follows.

Definition 5 (unordered top-quality planning problem) *Given a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$ and a natural number q, find a set of plans $P \subseteq \mathcal{P}_\Pi$ such that P is a solution to the quotient top-quality planning problem under the equivalence relation $U(\Pi)$.*

Note that while the solution to the top-quality planning problem can be obtained from a solution to the unordered top-quality planning problem, using a simple algorithm that generates all possible valid reordering for each plan in the solution, this is not the focus of current work. Focusing on the unordered top-quality planning problem allows us to generate reorderings of the same plan only if and when these reorderings are actually needed.

4 Computation of Top-quality Plans

In order to obtain a solver for the computational problem specified above, we take an approach similar to Katz *et al.* (2018), and iteratively generate plans using an existing cost-optimal planner, and construct planning tasks with a reduced set of plans, by forbidding exactly the plans found so far. In contrast to Katz *et al.* (2018), we forbid not only a specific plan, but also all its possible reorderings. In order to achieve that, we thus instead of forbidding plans as sequences of actions, forbid plans as multisets. To be able to do that, we need to come up with a reformulation of a planning task that forbids all plans with the exact number of appearances for each action. Similar reformulation was recently suggested by Anonymous (2019) for diverse planning. The reformulation can forbid a single multiset, and thus for a set of plans, the union of their multisets was forbidden in each consecutive iteration. That way, possibly additional plans were forbidden. For diverse planning, that did not pose a problem. In our case, however, we need to ensure that we forbid *exactly* the set of plans that were previously found. For that, in what follows, we adapt the reformulation of Anonymous (2019) accordingly.

Alternatively, we can use the reformulation of Anonymous (2019) directly, creating a sequence of planning tasks, similarly to the way it was done in top-k planning (Katz *et al.* 2018). This poses two problems: the reformulated planning task size grows fast with each iteration, and, as in the iterative top-k planner, the mapping between the reformulated and original actions must be constantly maintained.

In this work, at each iteration we reformulate the original planning task to forbid all plans found so far. In this case, we do not need to maintain the mapping between the reformulated and original actions and keep the reformulated task size smaller. In the rest of this section we adapt the definition of Anonymous (2019) to a set of plans (as multisets), present an algorithm that exploits the adapted definition to derive top-quality solutions, and prove its soundness and completeness. We start by presenting the definition.

4.1 Forbidding a Plan as a Multiset of Actions

Slightly simplifying the definition of Anonymous (2019), we present a task reformulation that ignores orders between actions in a plan and thus also forbids all possible reorderings of a given plan, as well as all subplans.

Algorithm 1 Iterative top-quality planning.

Input: Planning task Π, quality bound q
  $P \leftarrow \emptyset$
  $\Pi' \leftarrow \Pi$
  while True **do**
    $\pi \leftarrow$ optimal plan to Π'
    if $cost(\pi) > q$ **then**
      break
    end if
    $P \leftarrow P \cup \{\pi' \mid \pi' \text{ is symmetric to } \pi\}$
    $\Pi' \leftarrow \Pi^-_P$ according to Definition 7
  end while
  return P

Definition 6 *Let $\langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$ be a planning task and π be a plan. The task $\Pi^-_\pi = \langle \mathcal{V}', \mathcal{O}', s'_0, s'_\star \rangle$ is defined as follows.*

- $\mathcal{V}' = \mathcal{V} \cup \{\overline{v}\} \cup \{\overline{v}_o \mid o \in \pi\}$, *with $\overline{v}$ being a binary variable, and $dom(\overline{v}_o) = \{0, \dots, m_o\}$, where m_o is the number of occurences of o in π,*
- $\mathcal{O}' = \{o^e \mid o \in \mathcal{O} \setminus \pi\} \cup \bigcup_{i=0}^{m_o} \{o^f_i \mid o \in \pi\}$, *where*
  $pre(o^e) = pre(o), \quad eff(o^e) = eff(o) \cup \{\langle \overline{v}, 0 \rangle\}$,
  $pre(o^f_i) = pre(o) \cup \{\langle \overline{v}_o, i \rangle\}$,
  for $0 \leq i < m_o$, $eff(o^f_i) = eff(o) \cup \{\langle \overline{v}_o, i{+}1 \rangle\}$,
  $eff(o^f_{m_o}) = eff(o) \cup \{\langle \overline{v}, 0 \rangle\}$, *and*
  $cost'(o^e) = cost'(o^f_i) = cost(o)$, $0 \leq i \leq m_o$,
- $s'_0[v] = s_0[v]$ *for all* $v \in \mathcal{V}$, $s'_0[\overline{v}] = 1$, *and* $s'_0[\overline{v}_o] = 0$ *for all* $o \in \pi$, *and*
- $s'_\star[v] = s_\star[v]$ *for all* $v \in \mathcal{V}$ *s.t.* $s_\star[v]$ *defined, and* $s'_\star[\overline{v}] = 0$.

The semantics of the reformulation is as follows. The variable $\overline{v}$ starts from the value 1 and switches to 0 when an action is applied that is not from plan π treated as a multiset. Once a value 0 is reached indicating a deviation from plan π, it cannot be switched back to 1. The finite-domain variables $\overline{v}_o$ encode the number of applications of the action o. The actions o^f_i are copies of the action o in π, counting the number of applications of o, as long as the number is not higher than the number of times it appears in π. Once the number of applications exceeds m_o, $\overline{v}$ is set to 0.

4.2 Forbidding Multiple Plans Exactly

In order to forbid multiple plans, the greedy approach of Anonymous (2019) forbids the superset of these plans by taking a superset of the multisets representing the plans. In our case, when optimality is required, we cannot follow the same approach. Instead, we present a reformulation that forbids exactly these plans and their subplans, and the possible reorderings. Our reformulation extends the one in Definition 6, by introducing a binary variable for each plan, encoding whether the plan was deviated from.

Definition 7 *Let $\langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$ be a planning task, P be a set of plans, and $\mathcal{O}_P = \{o \mid o \in \pi, \pi \in P\}$. The task $\Pi^-_P = \langle \mathcal{V}', \mathcal{O}', s'_0, s'_\star \rangle$ is defined as follows.*

- $\mathcal{V}' = \mathcal{V} \cup \{\overline{v}_\pi \mid \pi \in P\} \cup \{\overline{v}_o \mid o \in \mathcal{O}_P\}$, *with* $\overline{v}_\pi$ *being binary variables, and* $dom(\overline{v}_o) = \{0, \ldots, m_o\}$, *where* $m_o = \max_{\pi \in P}\{m_o^\pi\}$ *and* m_o^π *is the number of occurences of* o *in* π,
- $\mathcal{O}' = \{o^e \mid o \in \mathcal{O} \setminus \mathcal{O}_P\} \cup \{o_i^f \mid o \in \mathcal{O}_P, 0 \leq i \leq m_o\}$, *where*
  $pre(o^e) = pre(o)$, $\mathit{eff}(o^e) = \mathit{eff}(o) \cup \{\langle \overline{v}_\pi, 0\rangle \mid \pi \in P\}$,
  $pre(o_i^f) = pre(o) \cup \{\langle \overline{v}_o, i\rangle\}$,
  $\mathit{eff}(o_i^f) = \mathit{eff}(o) \cup \{\langle \overline{v}_o, i{+}1\rangle\} \cup \{\langle \overline{v}_\pi, 0\rangle \mid i = m_{op}^\pi\}$ *for* $0 \leq i < m_o$,
  $\mathit{eff}(o_{m_o}^f) = \mathit{eff}(o) \cup \{\langle \overline{v}_\pi, 0\rangle \mid \pi \in P\}$, *and*
  $cost'(o^e) = cost'(o_i^f) = cost(o)$, $0 \leq i \leq m_o$,
- $s_0'[v] = s_0[v]$ *for all* $v \in \mathcal{V}$, $s_0'[\overline{v}_\pi] = 1$ *for all* $\pi \in P$, *and* $s_0'[\overline{v}_o] = 0$ *for all* $o \in \pi$, *and*
- $s_\star'[v] = s_\star[v]$ *for all* $v \in \mathcal{V}$ *s.t.* $s_\star[v]$ *defined, and* $s_\star'[\overline{v}_\pi] = 0$ *for all* $\pi \in P$.

4.3 Using the Reformulation

Algorithm 1 exploits the reformulation in Definition 7 to find a solution to the top-quality planning problem. The algorithm incrementally finds the set P of top quality plans. Starting with the empty set $P = \emptyset$ and assuming $\Pi_\emptyset^- = \Pi$, we use an optimal planner iteratively to find an optimal plan π to the planning task Π_P^-. Once a plan is found, it is added to the set of found plans P. Then, the new reformulation Π_P^- is constructed for the next iteration. The algorithm stops when a plan π is generated such that $cost(\pi) > q$. Note that the algorithm results in a set P of sequential plans, with no two plans being reorderings of each other. Similarly to Katz *et al.* (2018), at each iteration, after the plan π was found, we generate plans that are symmetric (Shleyfman *et al.* 2015) to π and add these to the set P. Finally, since the first step results in an optimal plan, the quality can be defined relatively to the cost of the optimal plan rather than an absolute number.

Theorem 1 *Algorithm 1 is sound and complete for top-quality planning when using cost-optimal planners that find shortest (in the number of actions) cost-optimal plans.*

Proof: Let P be the set of plans returned by Algorithm 1 and let π_f be the plan found when the algorithm breaks. Since π_f is an optimal plan to Π_P^- and $cost(\pi_f) > q$, we need to show that Π_P^- forbids exactly the plans in $reord(P)$. For a plan $\pi \in P$, Π_P^- has a variable $\overline{v}_\pi$ that reaches its goal value only when the number of applications of some action exceeds the number of appearances of that action in π. Thus, π is not a plan for Π_P^-. Since Definition 7 treats plans as multisets, this is true also for all $\pi' \in reord(\pi)$.

Let $P_1, \ldots, P_n$ denote the sets of plans at the beginning of each algorithm iteration and let $\pi_1, \ldots, \pi_n = \pi_f$ be the optimal plans found by the algorithm in these iteration, with π_i being an optimal plan to $\Pi_{P_i}^-$. Let π be a plan for Π such that $cost(\pi) \leq q$. If $\pi \notin reord(P)$, there exists k such that π is a plan for $\Pi_{P_k}^-$, but not for $\Pi_{P_{k+1}}^-$. Let $P' = P_{k+1} \setminus P_k$ be the plans forbidden in $\Pi_{P_{k+1}}^-$ but not in $\Pi_{P_k}^-$. Then, there exists $\pi' \in P'$ such that $\mathrm{MS}(\pi) \subseteq \mathrm{MS}(\pi')$. If $\mathrm{MS}(\pi) = \mathrm{MS}(\pi')$, then $\pi \in P$ and we are done. Assume that $\mathrm{MS}(\pi)$ is a proper subset of $\mathrm{MS}(\pi')$. Note that π' is a reordering of a plan that is symmetric to π_k, which was the optimal plan found for $\Pi_{P_k}^-$. Assuming that our optimal planner finds shorter optimal plans before longer ones, a plan π for $\Pi_{P_k}^-$ would be found before π_k, contradicting the assumption that $\mathrm{MS}(\pi)$ is a proper subset of $\mathrm{MS}(\pi')$. □

	$q_m = 1.00$		$q_m = 1.05$		$q_m = 1.10$		$q_m = 1.20$	
Coverage	K-tq	tq	K-tq	tq	K-tq	tq	K-tq	tq
airport	7	**21**	7	**18**	6	**17**	6	**17**
blocks	16	**17**	16	**17**	10	**13**	8	**9**
data-network18	0	**1**	0	0	0	0	0	0
depot	2	2	2	2	0	**2**	0	**1**
driverlog	5	**9**	5	**9**	1	**7**	1	**4**
floortile11	0	**2**	0	**2**	0	0	0	0
ged14	5	**7**	5	**7**	5	**7**	5	**7**
gripper	1	**4**	1	**3**	0	**2**	0	**2**
logistics00	3	**16**	3	**13**	1	**10**	0	**6**
logistics98	0	**4**	0	**2**	0	**1**	0	0
miconic	18	**27**	18	**26**	11	**16**	10	**12**
movie	0	**1**	0	**1**	0	**1**	0	0
mprime	18	**19**	18	**19**	18	**19**	6	**11**
mystery	20	20	20	20	20	20	13	**15**
nomystery11	9	**13**	7	**11**	4	**8**	2	**5**
openstacks08	0	**2**	0	**2**	0	**2**	0	**2**
parcprinter08	6	**15**	5	**12**	5	**12**	5	**11**
parcprinter11	3	**11**	2	**8**	2	**8**	2	**7**
pegsol08	21	**23**	21	**23**	21	**22**	8	**17**
pegsol11	8	**13**	8	**13**	8	**12**	2	**5**
pipes-notank	5	**11**	5	**11**	3	**7**	1	**4**
pipes-tank	2	**4**	2	**4**	2	**4**	1	1
psr-small	37	**46**	26	**40**	22	**36**	16	**24**
rovers	3	**6**	3	**6**	2	**4**	0	**3**
satellite	2	**5**	2	**5**	1	1	0	**1**
scanalyzer08	4	**5**	4	**5**	4	4	3	3
scanalyzer11	1	**2**	1	**2**	1	1	1	1
spider18	5	5	3	**4**	0	0	0	0
storage	8	**14**	8	**14**	7	**11**	6	**7**
tetris14	1	**2**	1	**2**	1	**2**	0	**1**
tidybot11	5	**7**	2	**5**	1	**3**	1	1
tpp	4	**6**	4	**5**	2	**5**	2	**5**
transport08	6	**7**	1	1	1	1	0	0
transport14	0	**1**	0	0	0	0	0	0
trucks	1	**2**	1	**2**	0	**1**	0	0
visitall11	8	8	7	7	5	**6**	5	5
woodwork08	3	**8**	2	**6**	1	**4**	0	**2**
woodwork11	0	**3**	0	**1**	0	0	0	0
zenotravel	7	7	7	7	4	**5**	3	**4**
Sum other	25	25	23	23	21	21	18	18
Sum (1797)	269	**401**	240	**358**	190	**295**	125	**211**

Table 1: The coverage results comparing to top quality planning via top-k planning, for various quality bounds.

5 Experimental Evaluation

In order to evaluate the feasibility of our suggested approach for unordered top-quality planning, we have implemented our approach on top of the Fast Downward planning system (Helmert 2006). The experiments were performed on Intel(R) Xeon(R) CPU E7-8837 @2.67GHz machines, with

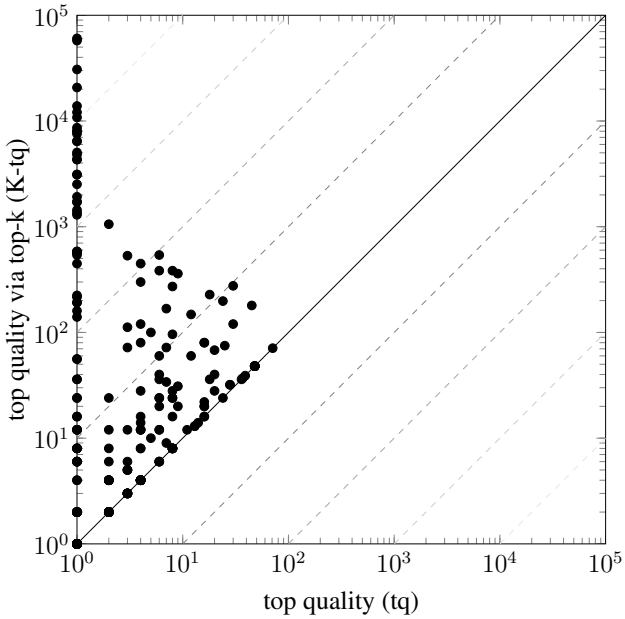

Figure 3: Per-task comparison of the solution encoding size.

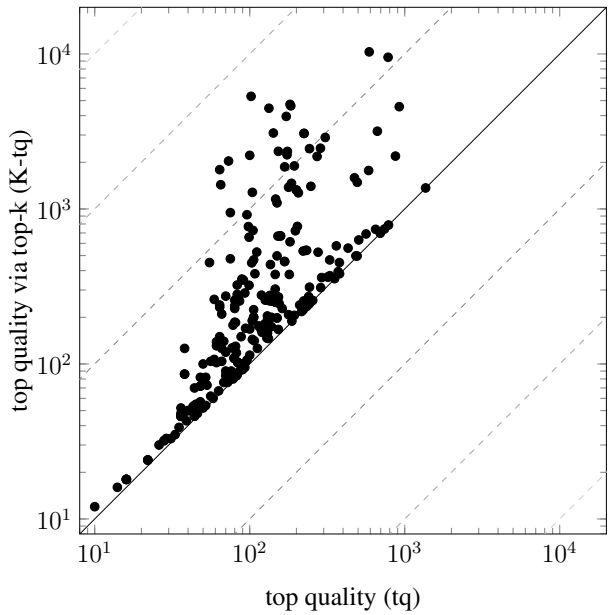

Figure 4: Final constructed task size in terms of the number of facts.

the time and memory limit of 30min and 2GB, respectively. The benchmark set consists of all STRIPS benchmarks from optimal tracks of International Planning Competitions (IPC) 1998-2018, a total of 1797 tasks in 64 domains. Our baseline for the comparison is a simple approach, using a top-k planner with a large k value, 10^9, stopping if a plan of quality above the bound was reached. We use *NaiveS*, the best perfoming configuration of the iterative top-k planner (Katz *et al.* 2018), that exploits both symmetries and plan reorderings. The purpose of setting k to a large number is to allow the top-k planner to exploit the entire 30min time interval. Among tasks solved, the largest number of plans found by the top-k planner was 60480. For tasks not solved, the maximal number of plans found by the top-k planner was 767501. Note that reading and writing such large amounts of plans is time consuming by itself. For each task, the quality bound is computed using the cost of the first found (optimal) plan, multiplied by a constant[2]. We experiment with four different quality bound multipliers, namely $q_m = 1.0$ (optimal plans only), 1.05, 1.1, and 1.2 of the optimal plan cost. For larger quality bounds, both approaches had low coverage, and thus we do not report these results. Note, q can be any natural number as mentioned in Definition 5.

Table 1 depicts the per-domain summed coverage, comparing our technique, tq, to the baseline, K-tq, for four quality bound multipliers. Each task gets a coverage of 1 if and only if the planner proved there is no other plan within quality bound, by either finding a plan above the bound or proving there are no other plans. Note first that out of the 64 domains, there are 18 domains where all optimal plans could not be found for any tasks, with any approach. There are 7 more domains where there is no difference in coverage between the baseline and our approach, for all tested quality bounds. These 25 domains are summarized in the *Sum other* row of Table 1. Out of the remaining 39 domains, the coverage never gets worse and it gets better (often significantly better) for at least one of the tested quality bounds. Extreme examples are AIRPORT, LOGISTICS00, and PSR-SMALL where the increase in coverage for some quality bounds is by 10 instances or more. Overall, there is a clear benefit of the suggested approach over the baseline.

Another benefit of our approach is a compact representation of the solution. Figure 3 shows a per-task comparison of the number of plans in the solution for each of the approaches. Finally, Figure 4 compares the reformulated task size of our approach to the baseline one. We compare the last generated task reformulation, for tasks solved by both approaches, for the quality bound multiplier $q_m = 1.0$. The task size is measured here by the number of facts, i. e., variable value pairs. While the larger tasks are not necessarily harder for classical planner, this is usually the case. Our experiments clearly show that our approach allows deriving task sizes of several orders of magnitude smaller than the baseline approach.

6 Conclusions and Future Work

In this work we have shown a way of obtaining all plans of bounded solution quality, representing plan reorderings implicitly and thus escaping the need for counting plans. We have presented a novel reformulation of a planning task that forbids exactly the set of given plans, their reorderings, and all subplans thereof. We have formally defined the problem of top-quality planning and have implemented a first planner for top-quality planning. The planner, exploiting the new reformulation, has empirically shown to perform significantly better than the straightforward approach of exploiting top-k

[2]This is not an overhead, as at least one optimal planner run needs to be performed anyway.

planners with a large bound k, as it is often done in practice.

For future work, one promising direction is exploring the use of top-quality instead of top-k planners in planning applications. Further, top-quality planners can be used to obtain solutions to diverse planning, when solution cost is also considered (Vadlamudi and Kambhampati 2016).

References

[Anonymous 2019] Anonymous. Reshaping diverse planning. 2019. Under review.

[Bäckström and Nebel 1995] Christer Bäckström and Bernhard Nebel. Complexity results for SAS$^+$ planning. *Computational Intelligence*, 11(4):625–655, 1995.

[Boddy *et al.* 2005] Mark Boddy, Johnathan Gohde, Tom Haigh, and Steven Harp. Course of action generation for cyber security using classical planning. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS 2005)*, pages 12–21. AAAI Press, 2005.

[Bryce 2014] Daniel Bryce. Landmark-based plan distance measures for diverse planning. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS 2014)* (2014), pages 56–64.

[2018] Tathagata Chakraborti, Kshitij P. Fadnis, Kartik Talamadupula, Mishal Dholakia, Biplav Srivastava, Jeffrey O. Kephart, and Rachel K. E. Bellamy. Visualizations for an explainable planning agent. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI 2018) - Demonstration Track*, pages 5820–5822. AAAI Press, 2018.

[2011] Alexandra Coman and Hector Muñoz-Avila. Generating diverse plans using quantitative and qualitative plan distance metrics. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI 2011)*, pages 946–951. AAAI Press, 2011.

[2006] Maria Fox, Alfonso Gerevini, Derek Long, and Ivan Serina. Plan stability: Replanning versus plan repair. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS 2006)*, pages 212–221. AAAI Press, 2006.

[2006] Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[2014] *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS 2014)*. AAAI Press, 2014.

[2018] Michael Katz, Shirin Sohrabi, Octavian Udrea, and Dominik Winterer. A novel iterative approach to top-k planning. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS 2018)*. AAAI Press, 2018.

[2018] Joseph Kim, Matthew E. Woicik, Matthew C. Gombolay, Sung-Hyun Son, and Julie A. Shah. Learning to infer final plans in human team planning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI 2018)*, pages 4771–4779. AAAI Press, 2018.

[2012] Tuan Anh Nguyen, Minh Binh Do, Alfonso Gerevini, Ivan Serina, Biplav Srivastava, and Subbarao Kambhampati. Generating diverse plans to handle unknown and partially known user preferences. *Artificial Intelligence*, 190:1–31, 2012.

[2015] Anton V. Riabov, Shirin Sohrabi, Daby M. Sow, Deepak S. Turaga, Octavian Udrea, and Long H. Vu. Planning-based reasoning for automated large-scale data analysis. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS 2015)*, pages 282–290. AAAI Press, 2015.

[2014] Mark Roberts, Adele E. Howe, and Indrajit Ray. Evaluating diversity in classical planning. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS 2014)* (2014), pages 253–261.

[2015] Alexander Shleyfman, Michael Katz, Malte Helmert, Silvan Sievers, and Martin Wehrle. Heuristics and symmetries in classical planning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015)*, pages 3371–3377. AAAI Press, 2015.

[2018] Maayan Shvo, Shirin Sohrabi, and Sheila A. McIlraith. An AI planning-based approach to the multi-agent plan recognition problem. In *Proceedings of the 31st Canadian Conference on Artificial Intelligence (CAI 2018)*, volume 10832 of *Lecture Notes in Computer Science*, pages 253–258. Springer-Verlag, 2018.

[2017] Silvan Sievers, Martin Wehrle, Malte Helmert, and Michael Katz. Strengthening canonical pattern databases with structural symmetries. In *Proceedings of the 10th Annual Symposium on Combinatorial Search (SoCS 2017)*, pages 91–99. AAAI Press, 2017.

[2016a] Shirin Sohrabi, Anton Riabov, and Octavian Udrea. Plan recognition as planning revisited. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*. AAAI Press, 2016.

[2016b] Shirin Sohrabi, Anton Riabov, Octavian Udrea, and Oktie Hassanzadeh. Finding diverse high-quality plans for hypothesis generation. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI 2016)*, pages 1581–1582, 2016.

[2018] Shirin Sohrabi, Anton V. Riabov, Michael Katz, and Octavian Udrea. An AI planning solution to scenario generation for enterprise risk management. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*, pages 160–167. AAAI Press, 2018.

[2016] Satya Gautam Vadlamudi and Subbarao Kambhampati. A combinatorial search perspective on diverse solution generation. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI 2016)*, pages 776–783. AAAI Press, 2016.