A New Approach to Plan-Space Explanation: Analyzing Plan-Property Dependencies in Oversubscription Planning

Rebecca Eifler,[1] **Michael Cashmore,**[2] **Jörg Hoffmann,**[1]
Daniele Magazzeni,[3] **Marcel Steinmetz**[1]
[1]Saarland University, Saarland Informatics Campus, Saarbrücken, Germany,
[2]University of Strathclyde, Computer and Information Sciences, Glasgow, UK,
[3]King's College London, Department of Informatics, London, UK,
{lastname}@cs.uni-saarland.de, {firstname.lastname}@strath.ac.uk, {firstname.lastname}@kcl.ac.uk

Abstract

In many usage scenarios of AI Planning technology, users will want not just a plan π but an explanation of the space of possible plans, justifying π. In particular, in oversubscription planning where not all goals can be achieved, users may ask why a conjunction A of goals is not achieved by π. We propose to answer this kind of question with the goal conjunctions B excluded by A, i. e., that could not be achieved if A were to be enforced. We formalize this approach in terms of plan-property dependencies, where plan properties are propositional formulas over the goals achieved by a plan, and dependencies are entailment relations in plan space. We focus on entailment relations of the form $\bigwedge_{g\in A} g \Rightarrow \neg \bigwedge_{g\in B} g$, and devise analysis techniques *globally* identifying all such relations, or *locally* identifying the implications of a single given plan property (user question) $\bigwedge_{g\in A} g$. We show how, via compilation, one can analyze dependencies between a richer form of plan properties, specifying formulas over action subsets touched by the plan. We run comprehensive experiments on adapted IPC benchmarks, and find that the suggested analyses are reasonably feasible at the global level, and become significantly more effective at the local level.

1 Introduction

AI plan generation technology serves to generate a plan π based on a world model. If the model and optimization objective reflect the real world and user preference with absolute accuracy, then π can be executed as-is. Yet in many usage scenarios, that is not so. Often, models are approximate, optimization objectives are highly complex and/or implicit in the heads of the human users, and bad plan decisions can be highly detrimental. A prominent example are space applications as discussed e. g. by Smith (2012), but arguably also many other applications ranging from Industry 4.0 to robot-aided disaster recovery. As Smith pointed out, planning should then be an iterative process in which the planning system suggests plan candidates, to be inspected and criticised by human users for iterative plan improvement.

As Smith also pointed out, plan explanation is a crucial step of such an iterative process, as a key means for user inspection. In particular, questions of the form "Why does the plan not achieve goal G?" or "Why don't you satisfy preference P?" need to be answered. Such answers require insights about the space of possible plans. We propose to address this by **plan-property entailments**, as in "Because achieving G would necessitate to either forego G' or use > 100 energy units". More formally, a plan property p is a Boolean function on plans (e. g. G true at end), and p entails q if all plans that satisfy p also satisfy q. A user question of the form "Why not p?" can then be answered in terms of those q entailed by p (e. g. G' false at end or energy usage > 100). We will refer to this as a **local explanation**, whereas a **global explanation** shows the graph of all plan-property entailments. The former makes sense in iterative planning as outlined, the latter makes sense when an overview of plan space is desirable. (For example, in simulated penetration testing (Boddy et al. 2005; Hoffmann 2015), an overview of the space of possible attacks may be useful.)

Our form of explanation falls into the class of *contrastive* explanations as discussed e. g. by Miller (2019). Previous work on local explanation, i. e., answering questions of the form "Why not p?", has suggested to generate a new plan π' that satisfies p, and answer the question based on comparing π and π' (Smith 2012; Fox, Long, and Magazzeni 2017). A weakness of this idea is that there may be differences between π and π' unrelated to p. Our approach replaces the *existential* answer generating a single alternative plan π' with a *universal* answer determining shared properties of *all* possible such alternatives. From this perspective, our proposal is a new, universal variant of contrastive plan explanation.

The concept of plan-property entailments is generic and in principle applicable to many AI Planning contexts. Here, we instantiate it for goal dependencies in oversubscription planning (OSP, e. g. (Smith 2004; Domshlak and Mirkis 2015)), where not all goals can be achieved. The plan properties we consider are goal-fact conjunctions $\bigwedge_{g\in A} g$ or negations $\neg \bigwedge_{g\in B} g$ thereof, and we identify **goal exclusion** entailments $\bigwedge_{g\in A} g \Rightarrow \neg \bigwedge_{g\in B} g$ stating that, if a plan achieves all goals in A, then it must forego at least one goal in B.

We spell out our framework for this form of plan properties and entailment relations. We introduce algorithms for computing local and global explanations, leveraging and extending recent nogood learning methods (Steinmetz and Hoffmann 2017b; 2017a). We show that the same framework can address a richer form of plan properties, namely **action-**

set properties defined as propositional formulas over atoms $\{A_1, \ldots, A_n\}$ asking whether an action subset A_i is touched by the plan. Action-set properties can be compiled into goal facts relatively easily, leading to an effective entailment analysis. We run comprehensive experiments on IPC benchmarks adapted to OSP as in previous work (Domshlak and Mirkis 2015; Katz et al. 2019), and on a collection of benchmarks we extended with action-set properties. We find that global explanations are reasonably feasible to compute, compared to OSP and to optimal classical planning. Computing local explanations is significantly easier.

Some related work will be discussed near the end of the paper (Section 7), to not interrupt the text flow.

2 Background

We consider the *finite-domain representation (FDR)* framework (Bäckström and Nebel 1995; Helmert 2009). An FDR task τ is a tuple $\tau = (V, A, c, I, G)$ where V is the set of **variables**, A is the set of **actions**, $c : A \to \mathbb{R}_0^+$ is the action **cost** function, I is the **initial state**, and G is the **goal**. A **state**, in particular I, is a complete assignment to V; G is a partial assignment to V; each action $a \in A$ has a **precondition** pre_a and an **effect** eff_a, both partial assignments to V. We will refer to variable-value pairs $v = d$ as **facts**, and we will identify partial variable assignments with sets of facts. An action a is **applicable** in a state s if $pre_a \subseteq s$. The outcome state $s[[a]]$ is like s except that $s[[a]](v) = eff_a(v)$ for those v on which eff_a is defined. The outcome state of an iteratively applicable action sequence π is denoted by $s[[\pi]]$.

An **oversubscription planning (OSP) task** is a tuple $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ like an FDR task but with two goals – the **hard** goal G^{hard} and the **soft** goal G^{soft}, assumed to be defined on disjoint sets of variables – as well as an additional **cost bound** $b \in \mathbb{R}_0^+$. A **plan** is an action sequence π whose summed-up cost is $\leq b$ and where $G^{\text{hard}} \subseteq I[[\pi]]$. Good plans achieve a "maximally valuable" subset of G^{soft}. The latter is usually defined based on goal-fact rewards. Here we assume instead that the user's preferences over G^{soft} are difficult to specify and/or elicitate, and planning is an iterative process as described by Smith (2012). We are concerned with the analysis of plan-property dependencies within G^{soft} to support that process.

We will also consider planning with non-reproducible resources, encoded as discrete FDR state variables. This can be viewed as a special case of our OSP formulation. In our experiments, among others we will use resource-constrained planning (RCP) benchmarks by Nakhost et al. (2012), which allow to control resource constrainedness (basically the ratio between available vs. required resource amounts).

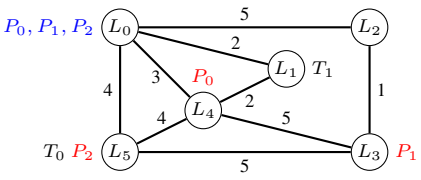

Figure 1: Running example from IPC NoMystery.

For illustration, we will use an RCP running example from IPC NoMystery, a transportation domain with fuel consumption. The variables encode truck and package positions, actions drive trucks or load/unload packages. Drive actions consume an amount of fuel (of the respective truck) depending on the road connection taken. As a concrete example, we will use the task with two trucks and three packages illustrated in Figure 1. Fuel consumptions are indicated at road segments. The packages are initially at L_0 (shown in blue); their goal locations are L_4, L_3,and L_5 (shown in red).

3 Goal-Property Dependencies in OSP

We next spell out our framework for plan properties, entailment relations between them, and the forms of explanations we aim at. We do so in FDR-based OSP as defined above, but in principle our definitions are generic and can be instantiated for arbitrary plan properties and planning frameworks. We will discuss the definitions from that perspective.

3.1 Plan Properties and Property Entailment

The plan properties we consider here are formulas over the soft goals:

Definition 1 (Plan Properties)**.** *Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, and Π its set of plans.*

*A **plan property** is a function $p_\phi : \Pi \to \{true, false\}$ where ϕ is a propositional formula over the atoms G^{soft}, and $p_\phi(\pi) = true$ iff ϕ evaluates to true under the truth value assignment where $g \in G^{\text{soft}}$ is true iff $g \in I[[\pi]]$. p_ϕ is **conjunctive** if ϕ has the form $\bigwedge_{g \in A} g$ or $\neg \bigwedge_{g \in B} g$.*

We identify p_ϕ with the characterizing formula ϕ. Our analyses are formulated for conjunctive plan properties (which through compilation can address arbitrary p_ϕ).

In general, a plan property can be any function mapping a task and an action sequence to a Boolean value. Examples are temporal plan trajectory constraints, deadlines, bounds on resource consumption, and the aforementioned action-set properties. To the extent that such properties can be compiled into goal facts, conjunctive plan properties can be used to analyze their dependencies. We will explore this possibility here with a compilation for action-set properties.

We formalize plan-property dependencies as entailments in the space of plans Π:

Definition 2 (Π-Entailment)**.** *Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, and Π its set of plans.*

*We say that $\pi \in \Pi$ **satisfies** a plan property ϕ, written $\pi \models \phi$, if $p_\phi(\pi) = true$. We denote by $\mathcal{M}_\Pi(\phi) := \{\pi \mid \pi \in \Pi, \pi \models \phi\}$ the subset of plans that satisfy ϕ. We say that ϕ Π-**entails** ψ, written $\Pi \models \phi \Rightarrow \psi$, if $\mathcal{M}_\Pi(\phi) \subseteq \mathcal{M}_\Pi(\psi)$.*

This definition views Π in the role traditionally taken by a knowledge base, identifying a set of "possible worlds" within which entailment over plan properties is considered. Observe that, given this, Π-entailment is more than standard entailment: $\phi \Rightarrow \psi$ implies that $\Pi \models \phi \Rightarrow \psi$, but not vice versa. Π-entailment captures entailments specific to the space of plans Π. For example, in our illustrative NoMystery task, say that all goals are soft, T_0 has initial fuel supply 13, and T_1 has no fuel. Then $\Pi \models at(P_0, L_4) \Rightarrow$

$\neg(at(P_1, L_3) \wedge at(P_2, L_5))$ because, if we achieve the goal for P_0, there is insufficient fuel to transport both other packages. If we set the initial fuel supply of T_0 to 16, on the other hand, then the knowledge base changes – Π becomes more permissive – and that entailment no longer holds.

Note that the definition of Π-entailment is agnostic to the specification of Π. The definition applies unchanged to arbitrary planning frameworks and plan sets Π.

Our primary focus here will be on goal exclusions:

Definition 3 (Goal Exclusions)**.** *Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, and Π its set of plans.*

*A **goal exclusion** is an entailment of the form $\Pi \models \bigwedge_{g \in A} g \Rightarrow \neg \bigwedge_{g \in B} g$. The entailment is **non-dominated** if there is no pair (A', B') where $A' \subseteq A$, $B' \subseteq B$, $(A', B') \neq (A, B)$, and $\Pi \models \bigwedge_{g \in A'} g \Rightarrow \neg \bigwedge_{g \in B'} g$. The entailment is **non-rhs-dominated** if there is no B' where $B' \subsetneq B$ and $\Pi \models \bigwedge_{g \in A} g \Rightarrow \neg \bigwedge_{g \in B'} g$.*

Goal exclusions are of interest in OSP as they reflect the detailed (soft-)goal trade-offs in the user's quest for a good plan. A non-dominated goal exclusion has subset-minimal A and B. This dominates entailments with larger A and/or B as it has a weaker left-hand side A (smaller conjunction) entailing a stronger right-hand side $\neg B$ (smaller disjunction, after moving the negation inside). If A is fixed, then only the right-hand side B needs be minimal. We will use non-dominated entailments to give more compact explanations.

3.2 Local and Global Explanations

As previously hinted, we propose to employ the concept of plan-property entailment for the purpose of giving local and global explanations of the plan space Π.

For local explanations, we assume a user question of the form "Why do you not achieve property ϕ?", which we answer with the set of plan properties ψ entailed by ϕ:

Definition 4 (Local Explanation (LE))**.** *Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, and Π its set of plans.*

*For a plan property ϕ, the **local explanation (LE)** for ϕ is the set $\{\psi \mid \Pi \models \phi \Rightarrow \psi\}$ of plan properties Π-entailed by ϕ. For $\phi = \bigwedge_{g \in A} g$, the **goal-exclusion LE** for ϕ is $\{\psi \mid \psi = \neg \bigwedge_{g \in B} g, \Pi \models \phi \Rightarrow \psi$ is non-rhs-dominated$\}$.*

Such an answer makes sense if the entailed properties ψ are undesirable. This is the case, in particular, for goal-exclusion LEs. In our example with T_0 fuel 13, the answer to $\phi = at(P_0, L_4)$ "Why do you not achieve the goal for P_0?" would be $\psi = \neg(at(P_1, L_3) \wedge at(P_2, L_5))$ "Because that would necessitate to forego the goal for either P_1 or P_2".

From a general perspective, plan properties here serve as an abstraction level at which to explain Π to a user. The underlying assumption is that Π is large and/or the mechanisms that generate Π are complex, so that an abstract form of explanation is needed. The abstraction level can be controlled through the number and granularity of plan properties. Given this, while here we simply talk about all formulas over soft-goal facts, it can make sense to instead fix a more specific set P of plan properties the user has a vested interest in (raising the new sub-problem how to choose P).

Note that, if the user question "Why do you not achieve property ϕ?" refers to a concrete given plan candidate π, then a simpler variant of local explanation is to return only those entailed ψ where $\pi \not\models \psi$, i. e., currently false plan properties that would become true when enforcing ϕ. This is easier to compute and yields smaller explanations. In our experiments, we will only use explanations as per Definition 4, yielding an upper bound on computational hardness and explanation size, and avoiding the bias incurred by a particular method for generating candidate plans π.

A canonical notion of global explanation arises directly from the above. Instead of showing the implications of one specific plan property, one can show all such implications:

Definition 5 (Global Explanation (GE))**.** *Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, and Π its set of plans.*

*Denote by $[\phi]_\Pi := \{\psi \mid \mathcal{M}_\Pi(\phi) = \mathcal{M}_\Pi(\psi)\}$ the Π-equivalence class of ϕ. The **global explanation (GE)** for τ is the strict partial order $\Rightarrow_\Pi$ over the classes $[\phi]_\Pi$ where $[\phi]_\Pi \Rightarrow_\Pi [\psi]_\Pi$ iff $[\phi]_\Pi \neq [\psi]_\Pi$ and $\Pi \models \phi \Rightarrow \psi$.*

The only somewhat non-obvious design decision here is to group plan properties into equivalence classes. That said, while this definition makes sense at the formal level, it is of doubtful practical value. There can be many plan properties (here: all propositional formulas over G^{soft}) and many equivalence classes (here: in the worst case, one class for every set of truth-value assignments to G^{soft}). This makes it questionable whether the GE can be processed by a user.

It therefore makes sense to focus on more limited forms of GEs, and to find ways to represent these more compactly. Here, we focus on non-dominated goal exclusions:

Definition 6 (Goal-Exclusion GE)**.** *Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, and Π its set of plans.*

*The **goal-exclusion GE** for τ is the strict partial order $\Rightarrow_\Pi$ over conjunctive plan properties where $\phi \Rightarrow_\Pi \psi$ iff $\Pi \models \phi \Rightarrow \psi$ is a non-dominated goal exclusion.*

Equivalence classes are not needed here because we consider only goal exclusions $\Pi \models \bigwedge_{g \in A} g \Rightarrow \neg \bigwedge_{g \in B} g$, so plan properties cannot entail each other. Our results on benchmarks suggest that the goal-exclusion GE is reasonably feasible to compute, and is often small.

Consider again our example, with T_0 fuel 13. Taking the freedom to notate goal facts by the respective packages, $\Pi \models P_0 \Rightarrow \neg(P_1 \wedge P_2)$, $\Pi \models P_1 \Rightarrow \neg(P_0 \wedge P_2)$, and $\Pi \models P_2 \Rightarrow \neg(P_0 \wedge P_1)$ are non-dominated goal exclusions. They are not the only ones though: $\Pi \models P_0 \wedge P_1 \Rightarrow \neg P_2$ also satisfies the definition, and so does any way of distributing the three packages across the left-hand vs. right-hand sides of the entailment. So the size of the goal-exclusion GE, in terms of the ordering relations it contains, is eight.

Note that this GE is equivalent to the statement "G^{soft} is not solvable as a whole, but each of its subsets is". Indeed, as we shall spell out next, for the special case of goal exclusions the GE is equivalent to the collection of minimal unsolvable goal subsets. We exploit this connection in our algorithms.

4 Computing Goal-Exclusion Explanations

We start with the algorithms for GEs, from which those for LEs follow easily.

4.1 Global Explanations

The special case of goal exclusions – though not the analysis of plan-property dependencies in general – boils down to the computation of **minimal unsolvable goal subsets (MUGS)**. A soft-goal set $G \subseteq G^{\text{soft}}$ is a MUGS if G cannot be achieved but every $G' \subsetneq G$ can:

Proposition 1 (GE from MUGS)**.** *Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, and Π its set of plans.*

Then $\Pi \models \bigwedge_{g \in A} g \Rightarrow \neg \bigwedge_{g \in B} g$, and that entailment is non-dominated, if and only if $A \cup B$ is a MUGS.

Proof. A Π-entailment $\Pi \models \bigwedge_{g \in A} g \Rightarrow \neg \bigwedge_{g \in B} g$ clearly holds iff $A \cup B$ is unsolvable. Non-dominated entailments result from set-inclusion minimal A and B, corresponding to the set-inclusion minimality of MUGS. □

We can hence represent goal-exclusion explanations more compactly, showing one MUGS G for each set of goal-exclusion entailments with $A \cup B = G$. In our example, the only MUGS is $\{at(P_0, L_4), at(P_1, L_3), at(P_2, L_5)\}$.

The question now is how to compute all MUGS. This can be done through a search over goal sets, that we refer to as **systematic weakening (SysW)**:

(1) the start node of the search is G^{soft};

(2) each search step selects an open node G, calls a planner to test whether G is solvable in τ, caches the result, and expands G if it is unsolvable;

(3) the children of a node G are those $G' \subset G$ where $|G'| = |G| - 1$.

Upon termination, the MUGS are those nodes G all of whose children are solvable.

Dually, **systematic strenghtening (SysS)** starts from $\emptyset$, with search steps expanding solvable nodes, and children adding one more goal fact. Upon termination, the MUGS can be easily obtained from the unsolvable search nodes.

In both algorithms, the worst-case number of planner calls is exponential in $|G^{\text{soft}}|$, i. e., the number of goals whose dependencies are being analyzed. Intuitively, SysW is better suited if solvable goal subsets are large (and thus are encountered early), while SysS is better suited if they are small (unsolvable subsets are encountered early). For illustration, in the example SysW expands the single search node G^{soft}, while SysS expands all solvable subsets before reaching G^{soft}.

Search enhancements are important for scalability. In particular, goal sets can be reached from the start node by permutations of goal-fact removal/addition steps. For effective results re-use, we assign goal sets unique integer IDs and fix an ascending expansion order. When expanding a node G and considering a possible child G', there are three possibilities: 1. $id(G') > id(G)$: G' not seen yet, insert into open list; 2. $id(G') < id(G)$ and result for G' cached: use that result; 3. $id(G') < id(G)$ and result for G' not cached: then the search was previously cut off above G', so we know that G' is solvable (SysW) respectively unsolvable (SysS).

We created synergy with recent nogood learning techniques, **conjunction learning** (Steinmetz and Hoffmann 2017b) and **trap learning** (Steinmetz and Hoffmann 2017a). These techniques refine dead-end detection methods (nogoods) based on the unsolvable states encountered in state space search on a planning task. Trap learning identifies unescapable non-goal regions of the state space. Conjunction learning enriches the critical-path heuristic h^C with more atomic conjunctions, and learns clauses by dead-end-state reduction. As the children tasks in our searches are closely related to their parents, the refined nogoods are likely to be useful still. So we **transfer** the nogoods along search paths, resulting in iteratively stronger and stronger nogoods.

For both forms of learning, the nogoods learned depend on the goal, so that only some of the nogoods remain valid for transfer in SysW where children remove goals. To identify that nogood subset, in conjunction learning we remember the subset G of goals proved to be unreachable by h^C in a learning step. Clause validity is preserved so long as at least one member of G is still present. In trap learning, the validity test checks, for every node of the trap, whether the node is still either mutex with the goal, or is still proved unsolvable by the heuristic function.

We extended conjunction learning to deal with OSP plan-cost bounds, by checking not whether $h^C(s) = \infty$ but whether $h^C(s)$ exceeds the remaining available cost. For trap learning, no straightforward extension exists, as trap nodes represent sets of states and thus would need to reason about bounds on remaining available cost. As trap learning is not the focus of our work, we leave this open for future research and use trap learning on RCP benchmarks only.

We acknowledge that the basic structure of SysW and SysS is simple, and relates to various prior works addressing conflict analysis in one or the other form. Our contribution here consists in assembling the algorithms in a form suited to our purposes, plus the described enhancements. We briefly discuss related work in Section 7.

4.2 Local Explanations

For local explanation, we are given a soft-goal subset A whose non-rhs-dominated entailments we wish to identify. This corresponds to fixing A and minimizing only B – which in turn corresponds to moving A into the hard goals and computing MUGS on the modified task.

Precisely, assume an OSP task $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$, with set of plans Π. Consider the modified task $\tau' := (V, A, c, I, G^{\text{hard}} \cup A, G^{\text{soft}} \setminus A, b)$. Clearly, we have the goal exclusion $\Pi \models \bigwedge_{g \in A} g \Rightarrow \neg \bigwedge_{g \in B} g$ iff B is unsolvable in τ'; and B is minimal in τ' iff the goal exclusion is non-rhs-dominated. So the MUGS in τ' yield exactly the goal-exclusion LE for A as per Definition 4.

Observe that τ' has fewer soft goals, so MUGS computation becomes easier, resulting in an advantage of local analysis over global analysis. That advantage grows with $|A|$, i. e., intuitively, with how specific the user question is. At the extreme end, $A = G^{\text{soft}}$ and both SysW and SysS on τ' contain a single search node testing solvability of $G^{\text{hard}} \cup G^{\text{soft}}$.[1]

[1] As previously indicated, given a concrete candidate plan π one could choose to restrict the answer to entailments $\neg \bigwedge_{g \in B} g$ where $\pi \models \bigwedge_{g \in B} g$. This corresponds to removing all g with $\pi \not\models g$ from G^{soft} in τ', further simplifying the computation.

5 Action-Set Properties

The analyses just described can, in principle, be used to analyze dependencies between arbitrary plan properties, so long as these can be compiled into goal facts. Given the well-known power of compilation in planning languages (e. g. (Gazen and Knoblock 1997; Nebel 2000; Edelkamp 2006; Palacios and Geffner 2009; Baier, Bacchus, and McIlraith 2009)), there is large potential in this idea. As an example, here we consider what we refer to as action-set properties:

Definition 7 (Action-Set Properties)**.** *Let $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ be an OSP task, Π its set of plans, and $A_1, \ldots, A_n \subseteq A$.*

*An **action-set property** for τ and $A_1, \ldots, A_n$ is a function $p_\phi : \Pi \to \{true, false\}$ where ϕ is a propositional formula over the atoms $A_1, \ldots, A_n$, and $p_\phi(\pi) = true$ iff ϕ evaluates to $true$ under the truth value assignment where A_i is $true$ iff π contains at least one action from A_i.*

As before, we identify action-set properties p_ϕ with the characterizing formulas ϕ. Arguably, action-set properties are practically relevant. They allow to express things like "objective x is covered by satellite y", "route x is not used", "passengers x and y ride in the same vehicle", if these are desirable but could be traded against other soft preferences. At the same time, the simple syntax of action-set properties lends itself to effective compilation, as follows.

Given τ, Π, and $A_1, \ldots, A_n$ as in Definition 7, to obtain a compiled task τ'

- introduce Boolean flags $isUsed_i$ that are initially $false$ and set to $true$ by any action from A_i;
- introduce formula-evaluation state variables and actions evaluating each p_ϕ based on the $isUsed_i$ flags (following (Gazen and Knoblock 1997; Nebel 2000)), setting Boolean flags $isTrue_\phi$ storing the outcome values;
- introduce a separate 1. *planning phase* vs. 2. *formula-evaluation phase*, and a switch action allowing to go from 1. to 2. if G^{hard} is satisfied.

Then the planning-phase prefixes in τ' are in one-to-one correspondence with Π, and given such a prefix π the evaluation phase in τ' can achieve $isTrue_\phi$ iff $p_\phi(\pi) = true$.

Now say that we want to analyze the dependencies across a given set P of action-set properties (e. g. possible undesirable consequences of not using route X). We are given τ, Π, and P; we want to identify dependencies of the form $\Pi \models \bigwedge_{\phi \in A} \phi \Rightarrow \neg \bigwedge_{\psi \in B} \psi$. With the above, this can be done by considering τ' with soft goals $\{isTrue_\phi \mid \phi \in P\}$, and identifying each $isTrue_\phi$ with ϕ in the outcome.

In the NoMystery domain action-set properties that can make sense are, for example, *uses* T_i (L_x, L_y) (truck T_i drives at least once from L_x to L_y or vice versa), *doesn't use* T_i (L_x, L_y) (the opposite), or *same truck* P_x P_y (both packages are delivered by the same truck). In our example task, say we fix the package destinations as hard goals, and we want to analyze the properties 1. uses T_0 (L_2, L_3); 2. same truck P_1 P_2; 3. uses T_0 (L_4, L_3); 4. same truck P_2 P_0; 5. doesn't use T_0 (L_0, L_5); 6. uses T_1 (L_5, L_4). Say initial fuel is 16 for T_0 and 7 for T_1. Computing the MUGS over the soft goals representing the six action-set properties, it turns out there are seven minimal unsolvable subsets of these properties, each of size three. A user could, for example, ask "Why do you not avoid the road $L_0 - L_5$ (which has a lot of traffic at the moment)?", translating into the question "Why do you not achieve property 5?". One of the non-rhs-dominated entailments of property 5 is $\neg$(property 2 $\wedge$ property 4), corresponding to the answer "Because if you don't use that road, then you cannot deliver all packages with a single truck."

We remark that, in a preliminary exploration, we implemented a compilation for LTL plan properties based on previous work (Edelkamp 2006; Baier, Bacchus, and McIlraith 2009). Our results are promising, yet indicate that algorithmic optimizations are needed to obtain good performance for complex properties. This remains a topic for future work.

6 Experiments

We implemented our approach in Fast Downward (FD) (Helmert 2006). We evaluate it, in turn, on IPC benchmarks modified for OSP planning, and on a selection of IPC benchmarks we extended with action-set properties.

The base planner called by our SysS and SysW algorithms on each search node runs forward search using h^{FF} (Hoffmann and Nebel 2001), optionally with conjunction or trap learning. The experiments were run on a cluster of Intel E5-2660 machines running at 2.20 GHz, with time (memory) cut-offs of 30 minutes (4 GB).

6.1 IPC-Based OSP Benchmarks

Following Katz et al. (2019), for every IPC benchmark task (V, A, c, I, G) with smallest known plan cost C as per planning.domains (Muise 2016), we obtained three OSP tasks by setting the cost bound to $b = x * C$ where $x \in \{0.25, 0.5, 0.75\}$. We used soft goals only, i. e., $G^{\text{soft}} = G$ and $G^{\text{hard}} = \emptyset$. For implementation reasons, we omitted tasks with ≥ 32 goals (which would be infeasible for MUGS analysis anyhow). We consider conjunction learning, but not trap learning as that cannot deal with OSP cost bounds.

IPC Global Explanations Figure 2 shows our data for global explanations, i. e., computing all MUGS. Note first the #MUGS data in the rightmost part of the table. As the data shows, the size of this global explanation is often small.

The rest of the figure focuses on computational performance. As a measure to compare against, we use optimal planning and OSP as reference points. The $h^{\text{LM-cut}}$ column gives coverage for A* with $h^{\text{LM-cut}}$ (Helmert and Domshlak 2009) run on the original IPC instance without a cost bound, as a comparison to solvable optimal planning. The OSP column gives coverage for the most recent OSP planner (Katz et al. 2019). It is expected that our algorithms, solving a more complex problem, will perform worse than the reference points. The question is, *how much worse?*

As a short summary of the answer provided by Figure 2 to that question, compared to optimal planning, with the smallest cost bound $x = 0.25$ our analysis is actually more effective. But that changes for larger cost bounds where the base planner's search space is larger and accordingly our analysis

	Reference Coverage				Systematic Strengthening/Weakening Coverage, x =												Search Tree Fraction, x =						#MUGS, x =					
	$h^{\text{LM-cut}}$	OSP			0.25				0.5				0.75				0.25		0.5		0.75		0.25	0.5	0.75	0.25	0.5	0.75
		x =			SysS		SysW		SysS		SysW		SysS		SysW		Sys		Sys		Sys							
domain		0.25	0.5	0.75	h^C		h^C		h^C		h^C		h^C		h^C		S	W	S	W	S	W	average			max		
agricola (20)	0	0	0	0	**20**	**20**	**20**	**20**	12	**13**	11	**13**	**2**	1	**2**	1	1.00	**0.50**	1.00	**0.50**	1.00	**0.50**	1.0	1.0	1.0	1	1	1
airport (50)	28	28	24	22	25	**35**	24	34	20	**21**	19	**21**	**20**	16	**20**	16	**0.60**	0.81	0.87	**0.73**	1.00	**0.61**	3.8	2.0	1.4	16	5	4
barman (34)	4	18	11	4	**18**	**18**	15	**18**	**11**	4	10	4	**8**	3	7	4	**0.57**	0.94	**0.88**	**0.88**	1.00	**0.50**	6.9	4.2	2.5	10	6	4
blocks (35)	28	35	28	21	**35**	**35**	29	**35**	25	**29**	22	**29**	18	**26**	18	**26**	**0.15**	0.97	**0.35**	0.95	0.80	**0.64**	11.0	12.4	13.7	59	40	57
childsnack (20)	0	2	0	0	0	**4**	0	**4**	0	0	0	0	0	0	0	0	**0.34**	0.98	-	-	-	-	16.8	-	-	20	-	-
data-network (20)	12	13	13	13	19	**20**	19	**20**	17	**18**	16	**18**	13	**17**	13	15	**0.72**	0.73	0.79	**0.70**	0.91	**0.66**	2.1	1.8	1.5	4	3	3
depot (22)	7	16	11	7	15	**16**	12	**16**	7	9	7	**10**	**6**	3	**6**	3	**0.24**	0.96	**0.54**	0.90	0.89	**0.68**	8.3	7.0	6.5	29	13	15
driverlog (20)	13	15	13	10	**15**	**15**	**15**	**15**	11	**12**	11	**12**	9	**10**	9	**10**	**0.17**	0.98	**0.59**	0.86	0.87	**0.50**	8.1	16.1	11.1	18	45	21
elevators (50)	40	22	22	22	**49**	47	**49**	48	**44**	38	43	37	**43**	27	41	27	**0.35**	0.94	**0.67**	0.89	0.90	**0.67**	4.6	5.1	5.9	14	12	21
floortile (36)	13	18	6	2	**15**	8	6	8	**3**	2	2	2	**2**	**2**	**2**	**2**	**0.10**	0.99	**0.67**	0.80	0.96	**0.30**	316.2	137.0	45.5	622	321	54
freecell (80)	15	77	30	21	51	**76**	41	**76**	25	**30**	21	**30**	15	**18**	15	**18**	**0.31**	0.94	**0.62**	0.94	0.88	**0.76**	4.0	4.3	3.3	4	6	5
ged (20)	15	20	20	20	16	16	15	**20**	**15**	10	10	12	**10**	7	**10**	7	**0.25**	0.90	**0.47**	0.80	**0.58**	0.70	13.3	38.7	12.5	30	101	38
grid (5)	2	5	3	2	4	**5**	4	**5**	3	3	3	**4**	2	**3**	2	**3**	**0.54**	0.84	0.83	**0.75**	1.00	**0.54**	4.0	2.5	1.0	7	4	1
gripper (20)	7	11	8	8	**8**	5	6	5	**5**	4	**5**	4	**5**	3	**5**	3	**0.21**	0.98	**0.65**	0.88	0.96	**0.46**	783.5	228.0	156.0	3060	792	495
hiking (20)	9	19	14	13	19	**20**	19	**20**	14	16	13	**17**	**14**	11	13	10	0.81	**0.69**	0.83	**0.67**	1.00	**0.63**	1.8	1.7	1.0	2	2	1
logistics (60)	26	27	20	16	7	**15**	6	**15**	4	**6**	4	**6**	3	3	3	**4**	**0.35**	0.95	**0.73**	0.81	0.98	**0.73**	7.2	7.3	2.8	18	22	4
miconic (150)	141	97	66	55	**75**	66	60	64	**51**	42	47	43	**45**	35	**45**	36	**0.30**	0.92	**0.73**	0.82	0.95	**0.61**	81.3	38.2	18.8	452	285	98
mprime (35)	22	35	27	24	**35**	**35**	35	**35**	28	**35**	28	**35**	25	**35**	24	**35**	0.90	**0.59**	0.91	**0.59**	0.94	**0.59**	1.3	1.2	1.1	2	2	2
mystery (30)	12	29	27	21	**30**	**30**	**30**	**30**	**30**	**30**	29	**30**	21	**30**	20	**30**	0.89	**0.61**	0.90	**0.61**	0.93	**0.61**	1.3	1.2	1.1	2	2	2
nomystery (20)	14	20	14	10	**20**	**20**	16	**20**	10	**12**	10	**12**	**8**	**8**	**8**	**8**	**0.15**	0.98	**0.61**	0.92	0.87	**0.61**	20.2	18.5	5.8	61	47	13
openstacks (77)	47	63	56	52	**49**	**49**	37	43	**47**	45	29	39	**42**	**42**	24	35	**0.03**	0.99	**0.04**	0.99	**0.12**	0.98	15.3	14.9	10.3	25	44	23
organic-syn-s (13)	10	8	8	8	**8**	**8**	7	**8**	**8**	**8**	7	**8**	**7**	6	6	6	**0.19**	0.96	**0.21**	0.95	**0.28**	0.91	5.3	7.3	8.3	12	28	36
parcprinter (26)	24	26	22	18	10	10	12	**14**	10	10	10	**14**	10	10	10	**12**	**0.44**	0.98	**0.61**	0.95	**0.73**	0.85	5.6	7.5	4.1	14	24	8
parking (40)	5	25	5	0	12	**17**	5	12	0	**1**	0	**1**	0	0	0	0	**0.02**	1.00	**0.27**	0.91	-	-	63.9	31.0		112	31	-
pathways (30)	5	5	4	4	5	**7**	5	**7**	4	**5**	4	**5**	**4**	**4**	**4**	**4**	**0.41**	0.86	**0.72**	0.78	0.91	**0.70**	11.3	3.8	1.8	50	10	3
pegsol (2)	2	2	2	2	0	0	**2**	**2**	0	0	**2**	**2**	0	0	**2**	**2**	-	-	-	-	-	-	7.0	23.5	64.0	8	41	122
pipesworld-nt (50)	17	45	30	23	42	**46**	38	**46**	**26**	25	24	**26**	**21**	15	20	15	**0.31**	0.94	**0.68**	0.83	0.88	**0.66**	5.0	5.6	4.3	13	31	24
pipesworld-t (50)	12	33	20	16	23	39	20	**40**	14	**18**	13	17	11	**13**	9	11	**0.35**	0.95	**0.63**	0.86	0.88	**0.65**	4.0	4.2	3.2	12	15	12
psr-small (50)	49	50	50	50	48	48	49	**50**	47	47	48	**49**	46	46	**48**	**48**	0.76	**0.63**	0.94	**0.55**	0.97	**0.47**	4.0	2.7	2.0	20	13	11
rovers (40)	8	16	8	6	**12**	**12**	**12**	**12**	**7**	**7**	**7**	**7**	**7**	5	**7**	5	**0.31**	0.95	**0.74**	0.84	0.92	**0.55**	21.0	11.4	4.3	95	35	12
satellite (36)	7	9	7	6	9	**12**	7	**12**	6	6	6	**7**	**6**	5	**6**	**6**	**0.12**	0.98	**0.49**	0.94	0.90	**0.68**	31.1	23.3	20.5	136	76	56
scanalyzer (40)	23	26	21	21	9	**17**	9	15	**9**	**9**	**9**	**9**	**9**	5	**9**	**9**	**0.22**	0.99	**0.53**	0.86	**0.75**	0.83	35.2	37.8	62.6	106	71	130
snake (17)	6	17	17	14	**7**	**7**	6	**7**	**3**	**3**	**3**	**3**	**3**	1	**3**	1	**0.16**	0.93	**0.32**	0.86	**0.58**	0.74	11.4	20.3	38.3	17	25	59
sokoban (50)	49	50	49	45	**50**	**50**	49	**50**	**48**	43	**48**	43	**44**	31	**44**	29	**0.59**	0.85	0.86	**0.72**	0.96	**0.49**	4.9	4.0	2.2	56	32	14
spider (12)	11	12	12	12	**6**	**6**	1	1	**5**	1	0	0	0	0	0	0	0.00	1.00	-	-	-	-	21.0	32.4	2.3	36	57	-
storage (30)	15	20	17	15	**18**	**18**	**18**	**18**	**16**	**16**	**16**	**16**	**15**	14	**15**	14	**0.57**	0.83	0.85	**0.75**	0.98	**0.57**	5.0	4.3	8.9	15	11	5
termes (20)	6	20	16	12	**20**	11	13	10	**15**	2	3	3	**7**	0	1	0	**0.28**	0.80	**0.53**	**0.53**	-	-	3.8	6.0	-	10	18	24
tetris (17)	6	17	14	11	8	**9**	7	7	**5**	3	3	3	**3**	2	**3**	2	**0.23**	0.98	0.81	**0.77**	0.97	**0.41**	38.2	43.8	7.3	79	89	11
tidybot (40)	23	40	38	32	**40**	**40**	39	**40**	**32**	31	25	28	**14**	13	8	**14**	**0.38**	0.92	**0.47**	0.91	**0.69**	0.79	3.1	3.3	2.9	4	6	6
tpp (30)	7	9	7	6	8	**9**	8	**9**	**7**	**7**	6	**7**	**6**	5	**6**	5	**0.36**	0.89	**0.66**	0.82	0.96	**0.66**	8.0	8.9	4.2	33	25	11
transport (70)	23	24	24	24	**42**	**42**	37	**42**	**34**	**34**	**34**	33	31	23	**33**	26	**0.37**	0.94	**0.59**	0.89	0.73	**0.69**	5.3	4.1	2.7	22	10	10
trucks (30)	10	12	8	6	12	**14**	9	**14**	**7**	**7**	6	**7**	**6**	4	6	4	**0.18**	0.98	**0.68**	0.89	0.93	**0.62**	30.2	17.4	7.7	137	32	21
visitall (14)	14	14	14	13	**13**	**13**	10	10	**10**	**10**	8	**10**	6	6	7	**8**	**0.20**	0.93	**0.36**	0.91	0.79	**0.75**	99.6	166.0	115.1	313	602	466
woodworking (35)	29	25	12	10	**23**	**23**	13	15	**9**	**9**	7	**9**	**5**	**5**	**5**	**5**	**0.02**	1.00	**0.22**	0.94	0.72	**0.52**	198.0	95.1	19.8	532	182	28
zenotravel (20)	13	13	10	8	**13**	**13**	12	**13**	**10**	9	8	9	8	**9**	8	**9**	**0.34**	0.94	**0.65**	0.88	0.86	**0.64**	10.5	6.7	2.6	36	31	4
Sum (1517)	828	1088	828	705	963	**1026**	846	1005	**714**	690	637	694	**580**	522	547	528												

Figure 2: Global explanation results on IPC benchmarks modified for oversubscription planning (OSP), with cost bounds set to x times optimal cost. Reference: A* with $h^{\text{LM-cut}}$ on original task, and OSP planning (Katz et al. 2019). SysS and SysW with vs. without conjunction learning h^C. Search tree fraction: fraction of worst-case search tree explored. #MUGS: average/maximum number of MUGS, i. e., global explanation size. Best performance in each part shown in **boldface**.

becomes harder. Taking the per-domain best of our four algorithm configurations, for $x = 0.25$ we get equal or better coverage in 37 of the 45 domains, for $x = 0.5$ that number is 25, for $x = 0.75$ it is 16. Compared to OSP planning, these numbers are 29, 18, and 22. Overall, it seems fair to say that our analysis is reasonably feasible: in many domains, it is on par with the most closely related simpler problems.

The search tree fraction explored by SysS increases with larger cost bounds, which is expected as solvable goal subsets become larger. For the same reason, the search tree fraction explored by SysW decreases; but this does not outweigh the increased base planner effort. Conjunction learning helps mostly for small cost bounds and/or when using SysW.

Explanation Examples To complement the purely computational evaluation of our approach with a view on the explanations provided, we now discuss a number of example domains and instances. We consider $x = 0.75$, arguably the most interesting setting. We selected five domains to cover a range of intuitively different structure. From each of these domains, we selected an instance whose size was feasible for us to discuss here.

- Blocksworld (instance 5-0): The goal facts are 1. $on(a, e)$, 2. $on(b, d)$, 3. $on(d, c)$ and 4. $on(e, b)$. The MUGS are $\{1, 3\}$, $\{1, 4\}$, $\{2, 3, 4\}$. So, for example, the question "Why don't you stack a on e?" is answered with "Because then I can't achieve the goals for d and e."
- Elevators (opt2008 instance 04): The goals are 1. $at(p_0, n_1)$, 2. $at(p_1, n_3)$, 3. $at(p_2, n_4)$ and 4. $at(p_3, n_2)$. The MUGS are $\{1, 2, 4\}$, $\{1, 3\}$, $\{2, 3\}$, $\{3, 4\}$. So the question "Why don't you transport p_3?" is answered with "Because then I can't transport anybody else."
- Freecell (instance 01): The goals are 1. $home(clubs_2)$, 2. $home(spades_2)$, 3. $home(hearts_2)$ and 4. $home(diamonds_2)$. The MUGS are $\{1, 2, 3\}$, $\{1, 2, 4\}$, $\{1, 3, 4\}$, $\{2, 3, 4\}$. So the question "Why don't you finish clubs and spades?" is answered with "If I do that, then I cannot finish the other suits."

  An interesting aspect of this example is the symmetry in the MUGS. A more compact and more general explanation would be "It is only possible to finish two card suits.", making clear that clubs and spades are no exception. It should be possible to identify opportunities for such explanations automatically (which might also be relevant in other contexts (Yu et al. 2017; Lauffer and Topcu 2019)); this remains a topic for future work.
- Rovers (instance 04): The goal facts are 1. $soil\_data(w_3)$, 2. $rock\_data(w_1)$, and 3. $image\_data(o_0, high\_res)$. The MUGS are $\{1, 3\}, \{2, 3\}$. So the question "Why don't you

take the image?" is answered with "Because then I can't take any of the samples."

- Woodworking instance 01(opt08): There are the goals 1. $colour(p_0, mauve)$, 2. $colour(p_1, green)$, 3. $surface\_condition(p_0, verysmooth)$, 4. $surface\_condition(p_1, smooth)$. Among the MUGS there are $\{1, 4, 5\}$, $\{2, 4, 5\}$. So the question "Why don't you grind p_0 and p_1?" is answered with "Because then I cannot paint them."

IPC Local Explanations For local explanations – entailments of soft-goal conjunctions $\bigwedge_{g \in A} g$ – we set up an experiment where we used only benchmark tasks with ≥ 6 goals (resulting in 37 non-empty domains), scaled question size $|A|$ from 1 to 5, and randomly selected 10 questions of each size. Figure 3 shows the data for SysW with conjunction learning at the most challenging cost bound 0.75. Averages are taken over instances where all questions were solved, as otherwise the results are distorted by large instances solved only for high values of $|A|$. As expected, computational effort and explanation size decrease with $|A|$.

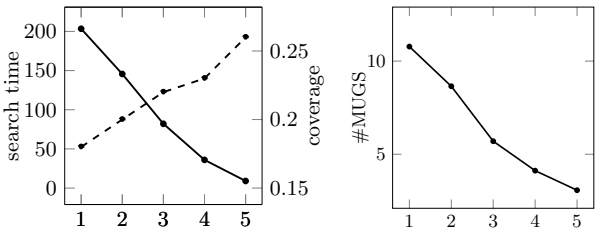

Figure 3: Local explanation results on IPC benchmarks, cost bound 0.75, for SysW as a function of $|A|$. Left: coverage, i. e., fraction of solved questions (dashed, right y-axis); and average runtime (solid, left y-axis). Right: average #MUGS.

These results are consistent across domains, and are much stronger in some cases. For example, while for $|A| = 1$ our coverage is higher than that of $h^{\text{LM-cut}}$ in only 2 domains here, for $|A| = 5$ that is so in 6 domains. Compared against OSP, the number of domains with higher coverage rises from 5 to 13. Local explanations are typically small even for domains where global explanations can be large as shown in Figure 2: the average number of MUGS for $|A| = 5$ is < 7 in all but 3 domains, and is < 15 except in Visitall (21.81).

6.2 Action-Set Properties

To evaluate the use of our framework with more complex plan properties, we experimented with the compilation of action-set properties as per Section 5. We extended four domains with action-set properties: NoMystery, Rovers, and TPP as per (Nakhost, Hoffmann, and Müller 2012), with controlled resource constrainedness (cf. Section 2); plus the Blocksworld as an intuitively differently structured domain. In NoMystery, the action-set properties are as in our illustrative example. In Rovers, the properties ask which rover or camera is used for which observation. In TPP, they ask which road segments are used, and which goods are bought at which markets. In Blocksworld, we include two hands and the properties ask which hand is used for which blocks. We set the original goal facts as hard goals, so that the analysis determines exclusion relations over conjunctions of action-set properties.

We encoded resource consumption into the FDR state variables, enabling the use of trap learning which turns out to be highly beneficial here. We generated benchmarks of sizes around the feasibility borderline, and we experimented with resource constrainedness $x \in \{1.0, 1.25, 1.5, 2.0\}$. For reference, we ran A* with $h^{\text{LM-cut}}$ as before, on all goal facts (original plus compiled action-set properties). Current OSP planners cannot handle hard goals, so we used our base planner with trap learning instead as a second reference point.

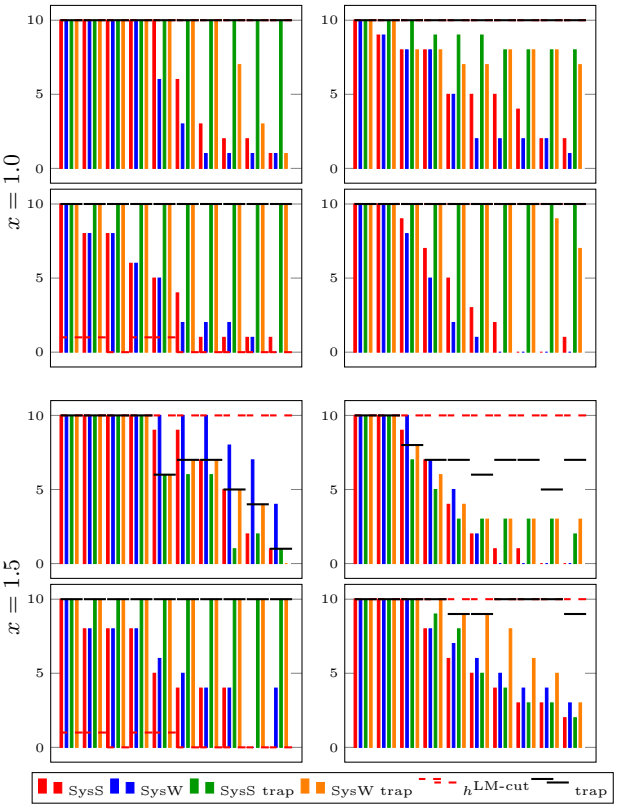

Figure 4: Coverage for global action-set property explanation at $x = 1.0$ and $x = 1.5$. Number of properties on x-axis, scaling from 1 to 10. Blocksworld top left, NoMystery top right, Rovers bottom left, TPP bottom right.

Figure 4 shows performance data for global explanation. We consider $x = 1.0$ and $x = 1.5$ as representatives of the overall picture. At $x = 1.0$, computing global action-set property explanations is feasible compared to the reference points. SysS with trap learning matches reference point coverage except for moderate losses in NoMystery. As more resources becomes available, similarly to before our analysis problem becomes harder to solve, and the coverage gap to the reference points increases. Rovers is an exception to this because, there, all pairs of action-set properties are unsolvable, which trap learning discovers quite effectively.

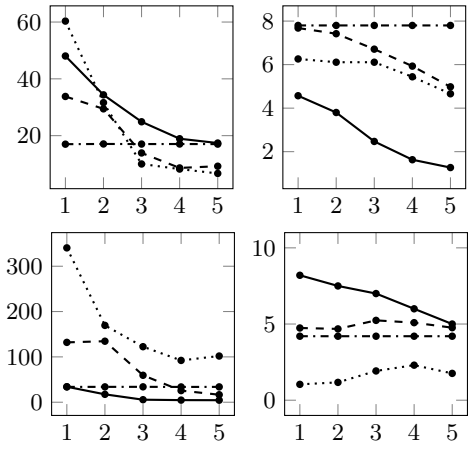

Figure 5: Local action-set property explanation at $x = 1.0$ (top) and $x = 1.5$ (bottom) SysS with trap learning. Average runtime (left) and average #MUGS (right) as a function of $|A|$. Blocksworld solid, NoMystery dashed, Rovers dash-dotted, TPP dotted.

Figure 5 shows performance data for local explanation. As before, this form of analysis becomes much easier as question size grows. runtime decreases significantly. Coverage (now shown in the figure to avoid cluttering the plots) increases accordingly. Explanation size, i. e., the number of MUGS, decreases as well (with the exception of Rovers as already pointed out).

7 Other Related Works

Various works relate to our computation of MUGS. In SAT and CP, minimal unsatisfiable cores (MUC, e. g. (Chinneck 2007; Laborie 2014)) are related to MUGS, yet rely on constraint-conflict analysis and are non-exhaustive, i. e., do not aim at finding *all* MUCs. Similarly, in the temporal planner IxTeT, an analysis of resource conflicts finds minimal critical sets (effects over-consuming a resource) (Laborie and Ghallab 1995). On the exhaustive side, problems and algorithms involving some form of solvability borderline within a lattice of problem variants have been considered since a long time (e. g. (de Kleer 1986; Reiter 1987)). In particular, the diagnosis framework by Grastien et al. (2011; 2012) can cast maximum solvable goal subsets as preferred diagnoses. That work uses SAT instead of a base planner, but an encoding into planning has been devised and ideas for search enhancements may be transferrable.

Some prior work (Yu et al. 2017; Lauffer and Topcu 2019) aims at suggesting goals to drop in oversubscribed situations, through conflict analyses related to finding a MUGS. At the technical level, both works are quite different from ours though as they use constraint-based problem encodings.

Prior work on other forms of plan-space explanation addressed unsolvable tasks, identifying minimal model changes resulting in solvability (Göbelbecker et al. 2010), or minimal differences between a solvable user model vs. unsolvable system model (Sreedharan et al. 2019).

We remark that our approach can be viewed as a form of domain/task analysis, or as a form of model checking applied to planning models. Both have been explored before, but addressing very different problems (e. g. (Fox and Long 1998; Rintanen 2000; Vaquero et al. 2013)).

Finally, to the reader versed in classical planning techniques, action-set properties may be reminiscent of disjunctive action landmarks (Karpas and Domshlak 2009; Helmert and Domshlak 2009), a technique used for computing heuristic functions. Disjunctive action landmarks correspond to atomic action-set properties. Propositional combinations other than conjunction (landmark collections) have not been considered though, and the previous purpose of disjuntive action landmarks is entirely different from ours.

8 Conclusion

Our approach analyses plan space in terms of plan properties and their dependencies. This naturally addresses challenges raised in previous work on iterative planning, and performance is reasonable in IPC benchmark studies. The approach can be easily generalized to almost arbitrary planning frameworks and plan-property languages, and we hope it will inspire other researchers too. To name just one example, plan properties and their dependencies might help in plan verbalization (Rosenthal, Selvaraj, and Veloso 2016), as an abstraction level (plan properties: "I did X") as well as a means for justification (dependencies: "I did not do Y because . . . ").

An interesting challenge is to answer deeper "why" questions regarding an entailment $\Pi \models p \Rightarrow q$. Possible ideas are to include additional plan properties elucidating the causal chain between p and q; or to find a minimal relaxation (superset) of the plan set Π for which p no longer entails q, thus elucidating the circumstances under which that entailment holds. A computational idea worth trying is to represent the plan set Π symbolically and use that representation to identify entailment relations. We aim to apply and evaluate our approach in concrete use cases such as simulated penetration testing and AUV mission control (Cashmore et al. 2014).

Acknowledgments

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-18-1-0245. Rebecca Eifler was also supported by the German Research Foundation (DFG) as part of CRC 248 (see perspicuous-computing.science). Daniele Magazzeni's group has also received support by EPSRC grant EP/R033722/1: Trust in Human-Machine Partnerships. Part of this work was performed while Jörg Hoffmann was visiting NASA Ames Research Center. We thank J. Benton, Minh Do, Jeremy Frank, and David Smith for insightful discussions.

References

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

Baier, J. A.; Bacchus, F.; and McIlraith, S. A. 2009. A heuristic search approach to planning with temporally extended preferences. *AI* 173(5-6):593–618.

Boddy, M.; Gohde, J.; Haigh, T.; and Harp, S. 2005. Course of action generation for cyber security using classical planning. In *Proc. ICAPS*, 12–21.

Cashmore, M.; Fox, M.; Larkworthy, T.; Long, D.; and Magazzeni, D. 2014. AUV mission control via temporal planning. In *Proc. ICRA*, 6535–6541.

Chinneck, J. W. 2007. *Feasibility and Infeasibility in Optimization: Algo- rithms and Computational Methods*. Springer-Verlag.

de Kleer, J. 1986. An assumption-based TMS. *AI* 28(2):127–162.

Domshlak, C., and Mirkis, V. 2015. Deterministic oversubscription planning as heuristic search: Abstractions and reformulations. *JAIR* 52:97–169.

Edelkamp, S. 2006. On the compilation of plan constraints and preferences. In *Proc. ICAPS*, 374–377.

Fox, M., and Long, D. 1998. The automatic inference of state invariants in TIM. *JAIR* 9:367–421.

Fox, M.; Long, D.; and Magazzeni, D. 2017. Explainable planning. In *Proc. IJCAI'17 Workshop on Explainable AI*.

Gazen, B. C., and Knoblock, C. 1997. Combining the expressiveness of UCPOP with the efficiency of Graphplan. In *Proc. ECP*, 221–233.

Göbelbecker, M.; Keller, T.; Eyerich, P.; Brenner, M.; and Nebel, B. 2010. Coming up with good excuses: What to do when no plan can be found. In *Proc. ICAPS*, 81–88.

Grastien, A.; Haslum, P.; and Thiébaux, S. 2011. Exhaustive diagnosis of discrete event systems through exploration of the hypothesis space. In *Proc. International Workshop on Principles of Diagnosis (DX)*.

Grastien, A.; Haslum, P.; and Thiébaux, S. 2012. Conflict-based diagnosis of discrete event systems: Theory and practice. AAAI Press.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proc. ICAPS*, 162–169.

Helmert, M. 2006. The Fast Downward planning system. *JAIR* 26:191–246.

Helmert, M. 2009. Concise finite-domain representations for PDDL planning tasks. *AI* 173:503–535.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302.

Hoffmann, J. 2015. Simulated penetration testing: From "Dijkstra" to "Turing Test++". In *Proc. ICAPS*.

Karpas, E., and Domshlak, C. 2009. Cost-optimal planning with landmarks. In Boutilier, C., ed., *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, 1728–1733. Pasadena, California, USA: Morgan Kaufmann.

Katz, M.; Keyder, E.; Winterer, D.; and Pommerening, F. 2019. Oversubscription planning as classical planning with multiple cost functions. In *Proc. ICAPS*, 237–245.

Laborie, P., and Ghallab, M. 1995. Planning with sharable resource constraints. In *Proc. IJCAI*, 1643–1649.

Laborie, P. 2014. An optimal iterative algorithm for extracting mucs in a black-box constraint network. In *Proc. ECAI*, 1051–1052.

Lauffer, N., and Topcu, U. 2019. Human-understandable explanations of infeasibility for resource-constrained scheduling problems. In *Proc. 2nd Workshop on Explainable Planning (XAIP'19)*.

Miller, T. 2019. Explanation in artificial intelligence: Insights from the social sciences. *AI* 267:1–38.

Muise, C. 2016. Planning.domains. In *ICAPS System Demonstrations and Exhibits*.

Nakhost, H.; Hoffmann, J.; and Müller, M. 2012. Resource-constrained planning: A Monte Carlo random walk approach. In *Proc. ICAPS*, 181–189.

Nebel, B. 2000. On the compilability and expressive power of propositional planning formalisms. *JAIR* 12:271–315.

Palacios, H., and Geffner, H. 2009. Compiling uncertainty away in conformant planning problems with bounded width. *JAIR* 35:623–675.

Reiter, R. 1987. A theory of diagnosis from first principles. *AI* 32(1):57–95.

Rintanen, J. 2000. An iterative algorithm for synthesizing invariants. In *Proc. AAAI*, 806–811.

Rosenthal, S.; Selvaraj, S. P.; and Veloso, M. M. 2016. Verbalization: Narration of autonomous robot experience. In *Proc. IJCAI*.

Smith, D. E. 2004. Choosing objectives in over-subscription planning. In *Proc. ICAPS*, 393–401.

Smith, D. 2012. Planning as an iterative process. In *Proc. AAAI*, 2180–2185.

Sreedharan, S.; Srivastava, S.; Smith, D.; and Kambhampati, S. 2019. Why couldn't you do that? explaining unsolvability of classical planning problems in the presence of plan advice. In *Proc. IJCAI*.

Steinmetz, M., and Hoffmann, J. 2017a. Search and learn: On dead-end detectors, the traps they set, and trap learning. In *Proc. IJCAI*.

Steinmetz, M., and Hoffmann, J. 2017b. State space search nogood learning: Online refinement of critical-path dead-end detectors in planning. *AI* 245:1–37.

Vaquero, T. S.; Silva, J. R.; Tonidandel, F.; and Beck, J. C. 2013. itsimple: towards an integrated design system for real planning applications. *Knowledge Engineering Review* 28(2):215–230.

Yu, P.; Williams, B. C.; Fang, C.; Cui, J.; and Haslum, P. 2017. Resolving over-constrained temporal problems with uncertainty through conflict-directed relaxation. *JAIR* 60:425–490.