Plan-Space Explanation via Plan-Property Dependencies: Faster Algorithms & More Powerful Properties

Rebecca Eifler* , **Marcel Steinmetz** , **Álvaro Torralba** and **Jörg Hoffmann**
Saarland University, Saarland Informatics Campus, Germany
{eifler, steinmetz, torralba, hoffmann}@cs.uni-saarland.de

Abstract

Justifying a plan to a user requires answering questions about the space of possible plans. Recent work introduced a framework for doing so via *plan-property dependencies*, where plan properties p are Boolean functions on plans, and p entails q if all plans that satisfy p also satisfy q. We extend this work in two ways. First, we introduce new algorithms for computing plan-property dependencies, leveraging symbolic search and devising pruning methods for this purpose. Second, while the properties p were previously limited to goal facts and so-called action-set (AS) properties, here we extend them to LTL. Our new algorithms vastly outperform the previous ones, and our methods for LTL cause little overhead on AS properties.

1 Introduction

As has been pointed out by Smith [2012] for space applications, the goal preferences for a planning task may be difficult to elicitate, calling for planning as an iterative process that suggests plan candidates for human inspection. Arguably this applies, for example, also to variants of production planning (Industry 4.0) and robot-aided disaster recovery. A key step in such a process is a variant of plan explanation answering user questions of the form "Why does the plan π you suggest not satisfy my preference p?". Eifler et al. [2020] recently introduced a framework addressing this explanation problem via **plan-property entailments**.

A plan property p is a Boolean function on plans, and p entails q if all plans that satisfy p also satisfy q. The answer to the above question could then be "Because achieving p would necessitate to either forego p' or use > 100 energy units". This is a form of contrastive explanation [Miller, 2019]. Some previous works [Smith, 2012; Fox *et al.*, 2017; Cashmore *et al.*, 2019; Krarup *et al.*, 2019] suggested to answer the question by comparing π to an alternative plan π' that satisfies p. In Eifler et al.'s framework, the answer consists instead of the properties *shared by all possible such* π'.

Eifler et al. [2020] (henceforth: "Eif20") assume a given set P of plan properties, and address oversubscription planning (OSP) [Smith, 2004; Domshlak and Mirkis, 2015] where not all of P can be satisfied. They identify all **exclusion dependencies** of the form $\bigwedge_{p\in X} p \Rightarrow \neg \bigwedge_{p\in Y} p$ where $X, Y \subseteq P$. Such dependencies are naturally suited to the desired Q/A pattern: "Why does the plan not satisfy the properties in X?" "Because then we would have to forego a property in Y".

Eif20 **compile** each plan property into a goal fact g. As $\bigwedge_{g\in X} g$ entails $\neg \bigwedge_{g\in Y} g$ iff $\bigwedge_{g\in X\cup Y} g$ is unsolvable, and as smaller property sets are stronger on each side of the entailment, the algorithmic problem thus boils down to computing all **minimal unsolvable goal subsets (MUGS)**.

Eif20 compute MUGS through a meta-search over goal subsets G, where each search node invokes a planner to decide solvability of G. They devise information flow across nodes through nogood transfer. Performance is reasonable in international planning competition (IPC) benchmarks constraining plan cost as in OSP [Katz *et al.*, 2019].

Our first contribution are two new algorithms for MUGS computation, as follows. (1) Run a symbolic search once to obtain a BDD representing the reachable state space, serving to implement the solvability tests efficiently. (2) Run a single explicit state-space search maintaining the set $\mathcal{G}_{\text{curr}}$ of solvable goal subsets found so far, adapting previous pruning and nogood learning techniques [Steinmetz and Hoffmann, 2017b] to detect states from which $\mathcal{G}_{\text{curr}}$ cannot be improved. Both (1) and (2) vastly outperform Eif20's methods.

Interestingly, we also obtain highly competitive results in a basic form of OSP, namely finding a largest solvable goal subset, which is solved as a side effect of finding all MUGS. Both (1) and (2) significantly outperform the most recent OSP solver [Katz *et al.*, 2019] on that problem.

Our second contribution consists in extending Eif20's framework to deal with more powerful plan properties. Eif20 consider goal facts as a simple canonical form of plan properties, and consider what they call **action-set (AS) properties** as a more complex form. AS properties p_ϕ are defined by a propositional formula ϕ over atoms $\{A_1, \ldots, A_n\}$, with A_i being true iff the plan uses at least one action $a \in A_i$. Eif20 devise a simple compilation of AS properties into goal facts.

Here, we tackle plan properties formulated in linear temporal logic, specifically LTL_f interpreted over finite traces [Baier *et al.*, 2009]. This strictly generalizes AS properties, and covers all but one of the preferences formulatable in PDDL3 [Gerevini *et al.*, 2009]. To compile LTL_f into

*Contact Author

goal facts (and thus stay within Eif20's algorithmic framework), we adopt previous techniques, producing the property automaton [Baier and McIlraith, 2006; Baier *et al.*, 2009] and compiling that automaton into the planning task [Edelkamp, 2006]. We empirically evaluate this method through experiments on a collection of IPC benchmarks extended with AS properties by Eif20. Formulating the same properties in LTL$_f$, we find that our more general solution causes little overhead. Formulating new LTL$_f$ plan properties beyond AS properties, we get worse but still reasonable scaling behavior.

Section 2 gives background on the planning formalism and Eif20's framework. Sections 3 and 4 describe our new algorithms for computing MUGS, Section 5 evaluates them. Section 6 describes our work on LTL$_f$. Some related work is discussed near the end of the paper in Section 7, to not interrupt the text flow. Section 8 concludes the paper.

2 Background

2.1 Oversubscription Planning

We consider a finite-domain variable variant of oversubscription planning (OSP) [Smith, 2004; Domshlak and Mirkis, 2015]. An **OSP task** is a tuple $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ 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**, G^{hard} (G^{soft}) is the **hard** (**soft**) **goal**, and $b \in \mathbb{R}_0^+$ is the **cost bound**. A **state** is a complete assignment to V; G^{hard} and G^{soft} are partial assignments to V, defined on disjoint sets of variables; each action $a \in A$ has a **precondition** pre_a and an **effect** eff_a, both partial assignments to V. We refer to variable-value pairs $v = d$ as **facts**, and we 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 $s[[a]](v) = \mathit{eff}_a(v)$ for those v on which eff_a is defined. The outcome state of an iteratively applicable action sequence π is denoted by $s[[\pi]]$. A **plan** is an action sequence π whose summed-up cost is $\leq b$ and where $G^{\text{hard}} \subseteq I[[\pi]]$.

We follow Eif20 in not defining a plan utility over G^{soft}. Instead, G^{soft} is a set of plan properties – including more general plan properties compiled into goal facts – and the analysis we provide identifies dependencies between these plan properties. The underlying assumption is that the user's preferences over G^{soft} are difficult to elicitate, and planning is an iterative process as described by Smith [2012].

We will also consider the basic OSP case where each soft goal yields the same positive reward, so that their number should be maximized. A plan π is **cardinality-optimal** if $G^{\text{soft}} \cap I[[\pi]]$ is maximal among all plans.

As an example, consider the OSP task illustrated in Figure 1, based on the IPC NoMystery domain where packages must be transported subject to limited fuel. The figure depicts the initial locations of trucks and packages (red), the goal locations of packages (green), and the amount of fuel consumed at each road connection. Truck T_0 initially has 5 fuel units, truck T_1 has 7. In addition to package location goals which are hard goals G^{hard}, we consider soft goals G^{soft} over more general plan properties. Examples that can be formulated as **action-set (AS)** properties as per Eif20 are "use the road connecting L_j and L_k, $use(T_i, L_j, L_k)$", "don't use the road connecting L_j and L_k, $not\_use(T_i, L_j, L_k)$", and "deliver two packages with the same truck $same(P_i, P_j)$".

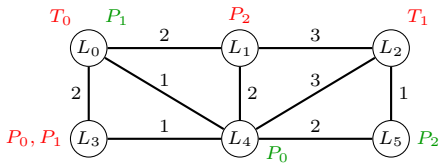

Figure 1: An illustrative NoMystery example.

2.2 Eifler et al.'s Framework

Eifler et al. [2020] (Eif20) introduce a framework for plan-property dependency analysis in OSP tasks, and algorithms for a specific instantiation of that framework. Here we consider only that instantiation, simplifying Eif20's concepts accordingly. We next outline the definitions, as well as Eif20's algorithms, so far as needed to understand our contribution.

Assume an OSP task $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$. A **plan property** is, in general, any function p mapping action sequences π in τ to Boolean values. Like Eif20's implementation, here we only consider p that can be compiled into goal facts, and we assume that the set of plan properties is represented by the set of soft goals G^{soft}. That is, we assume that a plan π satisfies a property p of interest iff the corresponding goal fact $g \in G^{\text{soft}}$ is true in $I[[\pi]]$. We are interested in conjunctions $\bigwedge_{g \in X} g$ of such plan properties, and specifically in **exclusion dependencies** where $\bigwedge_{g \in X} g$ "entails" $\neg \bigwedge_{g \in Y} g$.

Entailment here is intended as entailment *in the space of plans*. Clearly, goals will hardly entail other goals in the standard logical sense (in particular, goal facts never entail each other). However, it is often the case that all *plans* which satisfy X cannot satisfy Y. We denote by Π the set of plans for τ. We say that $\pi \in \Pi$ **satisfies** a formula ϕ over G^{soft}, written $\pi \models \phi$, if ϕ evaluates to true under the truth value assignment where $g \in G^{\text{soft}}$ is *true* iff $g \in I[[\pi]]$. 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 if $\mathcal{M}_\Pi(\phi) \subseteq \mathcal{M}_\Pi(\psi)$.

Applied to exclusion dependencies, this means that $\bigwedge_{g \in X} g$ Π-entails $\neg \bigwedge_{g \in Y} g$ if all action sequences in τ whose cost is within the bound b, that achieve G^{hard}, and that achieve all $g \in X$, do not achieve at least one $g \in Y$.

Our problem now is to compute all exclusion dependencies over G^{soft}. This corresponds to preparing, offline before iterative planning begins, answers to all questions of the form "Why does the plan not satisfy the properties in X?". To this end, observe that $\bigwedge_{g \in X} g$ Π-entails $\neg \bigwedge_{g \in Y} g$ iff $\bigwedge_{g \in X \cup Y} g$ is unsolvable in τ. Observe further that, in this case, the same is true for every $X' \supseteq X$ and $Y' \supseteq Y$, i. e., the exclusion dependency is strongest for set-inclusion minimal X and Y. Given these observations, the problem boils down to computing all **minimal unsolvable goal subsets (MUGS)**: all sets $G \subseteq G^{\text{soft}}$ where G cannot be achieved but every $G' \subsetneq G$ can. We will refer to this problem as **AllMUGS**. [1]

[1] Eif20 also define an online problem, addressing a specific user question asked during iterative planning. This means to solve AllMUGS in the modified task where X is moved from G^{soft} to G^{hard}.

Eif20 solve AllMUGS through a meta-search over soft-goal subsets G, either starting with G^{soft} and working downwards (systematic weakening, **SysW**), or starting with $\emptyset$ and working upwards (systematic strengthening, **SysS**). Each node in the meta-search invokes a planner to decide solvability. Towards efficiency, Eif20 employ recent nogood learning methods in the planner calls [Steinmetz and Hoffmann, 2017b; Steinmetz and Hoffmann, 2017a], and transfer the learned nogoods across nodes in the meta-search. This is always possible in SysS, and is possible in SysW whenever the goals a nogood depends on are still present.

In our example, say we consider the plan properties $1 : use(T_1, L_0, L_4)$, $2 : not\_use(T_0, L_0, L_1)$, $3 : not\_use(T_1, L_1, L_2)$, and $4 : same(P_0, P_1)$. Then the MUGS are $\{2, 3\}$ and $\{1, 2, 4\}$. So, for example, the answer to the user question "Why does the plan not drive T_1 from L_0 to L_4?" is "Because then either T_0 has to drive from L_0 to L_1, or P_0 and P_1 cannot be transported with the same truck."

3 Solving AllMUGS with Symbolic Search

The planner calls in Eif20's SysW and SysS meta-searches all solve the same planning task, except for the changing goal subset. Eif20 leverage this connection for nogood transfer, thus avoiding some redundant work. However, as all the solvable goal subsets can be read off one and the same state space, the redundant work can be avoided altogether. We introduce two methods to this end, based on (1) computing a BDD that represents all reachable states (in the present section), respectively (2) a single explicit state-space search maintaining a list of solvable goal subsets (in the next section).

Symbolic search is a well-known paradigm that uses Binary Decision Diagrams (BDDs) to compactly represent sets of states, often using exponentially less memory than their explicit enumeration. Successor-state generation can be directly implemented on the BDD representation, via BDD operations whose runtime depends only on the BDD size, not on the number of states represented. Symbolic search has been shown to be a powerful tool for state space exhaustion in model-checking [McMillan, 1993] and planning [Edelkamp and Kissmann, 2009; Torralba *et al.*, 2017].

Assembling this machinery for our purposes, the main difference to previous work is that we need to adhere to the cost bound. To this end, given an OSP task $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$, we run a symbolic forward uniform-cost search. We terminate when we reach the cost-bound b, when we reach a state satifying $G^{\text{hard}} \cup G^{\text{soft}}$, or when all reachable states have been generated. The result is a BDD, that we denote β_τ, which represents the set of states reachable in τ.

The BDD β_τ can be used to test solvability of each goal G in SysS and SysW efficiently: 1. produce a BDD β_G representing $\bigwedge_{g \in G} g$; 2. conjoin β_τ and β_G to obtain a BDD for $\beta_\tau \wedge \beta_G$; and 3. test whether that latter BDD represents the empty set. Operation 1 takes time linear in $|G|$, operations 2 and 3 take time linear in the size of β_τ times $|G|$. Given this fast test, using SysS vs. SysW hardly makes a performance difference anymore; in our experiments, we use SysS.

Note that, as a side effect, this algorithm solves cardinality-optimal OSP. A largest solvable goal subset G can be identified by maintaining flags alongside SysW/SysS. Given such G, one can easily extract a cardinality-optimal plan from the BDDs β_g produced by symbolic uniform-cost search for each g-cost value encountered. We implemented this technique as well, and will show that it yields interesting results.

4 AllMUGS Pruning and Nogood Learning

Our second new AllMUGS algorithm extracts all MUGS from a single explicit state-space search, exhausting all states reachable within the cost bound (or when a state is reached that satisfies $G^{\text{hard}} \cup G^{\text{soft}}$). During the exploration, we maintain reachability information for goal subsets, namely the set $\mathcal{G}_{\text{curr}}$ of set-inclusion maximal goal subsets reached so far. When search terminates, $\mathcal{G}_{\text{curr}}$ contains all maximal solvable goal subsets. The MUGS can be computed from this by first extending all $G \in \mathcal{G}_{\text{curr}}$ with one more soft-goal fact in all possible ways, and then keeping the set-inclusion minimal sets from the outome. We denote the result by $\text{MUGS}(\mathcal{G}_{\text{curr}})$. Note that, through the computation of $\mathcal{G}_{\text{curr}}$, the algorithm also computes cardinality-optimal plans as a side effect.

Pruning To avoid exhausting the entire state space, one can prune states s from which it is not possible to achieve "something new", i. e., from which one cannot reach any $G_{new} \subseteq G^{\text{soft}}$ with $G_{new} \not\subseteq G_{old}$ for any $G_{old} \in \mathcal{G}_{\text{curr}}$. Observe that this condition is satisfied iff s cannot reach any $G_{new} \in \text{MUGS}(\mathcal{G}_{\text{curr}})$. The question now is how to adopt known reachability approximations for this purpose.

Assume any admissible classical planning heuristic h. One can prune a state s with remaining budget b_s if (*) for all $G_{new} \in \text{MUGS}(\mathcal{G}_{\text{curr}})$, $h(s, G^{\text{hard}} \cup G_{new}) > b_s$. However, $\text{MUGS}(\mathcal{G}_{\text{curr}})$ may become large, and each call to h may incur a computational cost. So, with the pruning test conducted on every state during search, implemented naïvely this approach is bound to cause substantial overhead. Here we show how to implement the approach more effectively for two particular heuristic functions, namely the critical-path heuristics $h^{\max}$ [Haslum and Geffner, 2000] and h^C [Haslum, 2012]. Both allow to approximate (*) through a single computation of the heuristic, plus some additional processing:

- In a nutshell, $h^{\max}$ approximates the cost-to-go by the cost of reaching each fact individually. This allows to extract, from a single $h^{\max}$ computation, the set $P_{>b_s}$ of facts whose $h^{\max}$ estimate exceeds b_s. Then (*) is entailed if $G^{\text{hard}} \cap P_{>b_s} \neq \emptyset$; or if there exists $G \in \mathcal{G}_{\text{curr}}$ such that $G^{\text{soft}} \setminus P_{>b_s} \subseteq G$.
- The h^C heuristic generalizes $h^{\max}$ by taking into account the cost to *jointly* achieve the fact sets (the *conjunctions*) contained in the parameter C. From a single h^C computation, one can extract the set $P^C_{>b_s}$ of conjunctions whose h^C estimate exceeds b_s. Then (*) is entailed if $P^C_{>b_s}$ contains a subset of G^{hard}, i. e., ex. $c \in P^C_{>b_s}$ with $c \subseteq G^{\text{hard}}$; or if, for every $G_{new} \in \text{MUGS}(\mathcal{G}_{\text{curr}})$, $P^C_{>b_s}$ contains a subset of $G^{\text{hard}} \cup G_{new}$.

To make use of h^C, we need to obtain the set C of conjunctions to consider. To this end, we extend Steinmetz and Hoffmann's [2017b] nogood learning approach to our context.

	AllMUGS															Cardinality-Optimal OSP								
	$x = 0.25$					$x = 0.5$					$x = 0.75$					$x = 0.25$			$x = 0.5$			$x = 0.75$		
Domain	Eif	EifL	$h^{\max}$	h^C_L	Sym	Eif	EifL	$h^{\max}$	h^C_L	Sym	Eif	EifL	$h^{\max}$	h^C_L	Sym	Katz	$h^{\max}$	Sym	Katz	$h^{\max}$	Sym	Katz	$h^{\max}$	Sym
Agricola(20)	20	20	20	20	20	13	13	14	16	**20**	2	1	2	2	**19**	0	**20**	**20**	0	14	**20**	0	2	**19**
Airport(50)	25	**35**	33	34	26	19	21	**25**	22	22	19	16	**24**	16	21	28	**33**	26	24	**25**	22	22	**24**	21
Barman(34)	18	15	20	18	**25**	4	4	11	4	**18**	4	3	4	0	**14**	18	20	**25**	11	11	**18**	4	4	**13**
Blocksworld(35)	34	35	35	35	35	23	29	29	**30**	29	18	**26**	24	**26**	22	35	35	35	28	**29**	**29**	21	**24**	22
Childsnack(20)	0	4	2	**6**	**6**	0	0	0	0	**2**	0	0	**8**	6	2	2	2	**6**	0	0	**2**	0	**8**	2
Data-Network(20)	17	**20**	**20**	**20**	19	14	**18**	16	**18**	17	11	**17**	13	16	13	13	**20**	19	13	16	**17**	13	13	13
Depots(22)	12	16	16	16	**17**	7	9	**12**	10	**12**	4	3	**7**	3	**7**	16	16	16	11	**12**	**12**	7	7	7
Driverlog(18)	15	15	15	15	15	10	12	**14**	12	**14**	8	10	**12**	10	**12**	15	15	15	13	14	**15**	10	**12**	**12**
Elevators(50)	47	47	**50**	47	**50**	43	38	**44**	40	**44**	35	27	41	23	**43**	22	**50**	**50**	22	**44**	**44**	22	41	**43**
Floortile(36)	7	8	**18**	16	16	2	2	**6**	2	**6**	2	2	3	2	**5**	**18**	**18**	16	6	6	6	2	3	**5**
FreeCell(80)	42	76	**80**	77	76	16	30	**38**	29	30	14	18	**20**	18	**20**	77	**80**	76	30	**38**	30	**21**	20	20
GED(20)	16	16	**20**	**20**	16	15	10	**20**	19	15	10	7	**20**	10	11	**20**	**20**	16	**20**	**20**	15	**20**	**20**	11
Grid(5)	4	**5**	**5**	**5**	**5**	3	3	3	**4**	3	2	**3**	**3**	**3**	2	5	5	5	3	3	3	2	**3**	2
Gripper(14)	7	5	**11**	7	9	4	4	**7**	4	6	4	3	**7**	3	5	**11**	**11**	9	**8**	7	6	**8**	7	5
Hiking(20)	15	**20**	**20**	**20**	19	11	**16**	13	14	**16**	11	11	11	11	**15**	19	**20**	19	14	13	**16**	13	11	**15**
Logistics(32)	6	15	10	**16**	12	3	6	5	**7**	6	2	3	4	4	**5**	**27**	10	12	**20**	5	6	**16**	4	5
Miconic(150)	66	66	101	72	**105**	45	42	68	45	**71**	41	35	55	39	**61**	97	101	**105**	66	68	**71**	55	55	**60**
Mprime(35)	35	35	35	35	35	27	**35**	34	**35**	34	24	**35**	31	**35**	29	35	35	35	27	**34**	**34**	24	**31**	29
Mystery(30)	30	30	30	30	30	25	30	30	30	30	18	**30**	26	28	26	29	**30**	28	27	**30**	28	21	**26**	**26**
Nomystery(20)	20	20	20	20	20	10	12	**15**	13	14	8	8	10	8	**12**	20	20	20	14	**15**	14	10	10	**12**
Openstacks(77)	49	49	**63**	61	**63**	47	45	**61**	53	59	42	42	**60**	57	52	63	63	63	56	**61**	59	52	**60**	52
Org-syn-s(13)	8	8	**13**	12	9	8	8	**13**	11	8	7	6	**13**	8	8	8	**13**	9	8	**13**	8	8	**13**	8
Parcprinter(26)	10	10	**24**	22	10	10	10	**24**	18	10	10	10	**20**	16	10	**26**	24	10	22	**24**	10	18	**20**	10
Parking(40)	12	17	**26**	21	25	0	1	**5**	1	**5**	0	0	0	0	0	25	**26**	25	5	5	5	0	0	0
Pathways-nn(23)	5	7	6	7	**9**	4	5	5	5	**6**	4	4	4	4	**5**	5	6	**9**	4	5	**6**	4	4	**5**
Pegsol(2)	0	0	**2**	**2**	1	0	0	**2**	**2**	1	0	0	**2**	**2**	1	**2**	**2**	1	**2**	**2**	1	**2**	**2**	1
Pipesworld-nt (50)	41	**46**	**46**	**46**	45	23	25	**37**	27	31	21	15	**24**	15	22	45	**46**	45	30	**37**	31	23	**24**	22
Pipesworld-t (50)	23	39	37	40	**42**	12	18	22	18	**25**	19	13	16	12	**18**	33	37	**42**	20	22	**26**	16	16	**18**
PSR-small (50)	48	48	**50**	**50**	48	47	47	**50**	**50**	47	46	46	**50**	49	46	**50**	**50**	48	**50**	**50**	47	**50**	**50**	46
Rovers(31)	12	12	17	15	**19**	7	7	9	7	**14**	6	5	7	6	**14**	16	17	**19**	8	9	**14**	6	7	**14**
Satellite(19)	8	12	11	12	**15**	6	6	7	7	**12**	4	5	6	6	**8**	9	11	**15**	7	7	**12**	6	6	**8**
Scanalyzer(40)	9	17	**25**	23	21	9	9	**21**	9	9	9	5	**15**	5	9	**26**	25	21	**21**	**21**	9	**21**	15	9
Snake(17)	6	7	**17**	14	7	3	3	**14**	5	6	3	1	**10**	2	4	**17**	**17**	7	**17**	14	6	**14**	10	4
Sokoban(50)	50	50	50	50	50	46	43	**50**	43	48	40	31	**48**	31	44	50	50	50	49	**50**	48	45	**48**	44
Spider(12)	6	6	**12**	11	9	5	1	**12**	4	9	0	0	**10**	1	4	**12**	**12**	9	**12**	**12**	9	**12**	10	4
Storage(30)	18	18	**21**	20	**21**	16	16	**17**	16	**17**	15	14	**16**	15	15	20	**21**	20	17	17	17	15	**16**	15
Termes(20)	15	11	18	12	**20**	6	2	14	3	**18**	0	0	9	1	**15**	**20**	18	**20**	16	14	**18**	12	9	**15**
Tetris(17)	8	9	**17**	11	12	4	3	**12**	3	8	3	2	**8**	3	7	**17**	**17**	12	**14**	12	7	**11**	8	7
Tidybot(40)	40	40	40	40	40	28	31	37	32	**38**	13	13	26	16	**33**	40	40	40	**38**	37	**38**	32	26	**33**
TPP(30)	8	9	**13**	11	12	7	7	**8**	7	**8**	6	5	7	6	**8**	9	**13**	12	7	**8**	**8**	6	7	**8**
Transport(70)	37	42	43	42	**47**	34	34	34	34	**35**	27	23	24	24	**31**	24	43	**46**	24	34	**35**	24	24	**31**
Trucks(30)	12	14	15	16	**19**	6	7	9	9	**12**	5	4	8	4	**9**	12	15	**19**	8	9	**12**	6	8	**9**
Visitall(14)	13	13	**14**	**14**	13	9	10	**13**	10	10	6	6	**13**	9	7	**14**	**14**	13	**14**	13	10	**13**	**13**	7
Woodworking(35)	23	23	**35**	29	25	9	9	**25**	13	9	5	5	**15**	9	5	25	**35**	25	12	**25**	9	10	**15**	5
Zenotravel(20)	13	13	13	13	**14**	9	9	**12**	10	**12**	8	9	9	9	**10**	13	13	**14**	10	**12**	**12**	8	9	**10**
Sum (1517)	907	1026	**1189**	1123	1152	639	690	**917**	751	866	522	522	**745**	573	729	1088	**1189**	1147	828	**917**	865	705	**745**	727

	$x = 1.0$				$x = 1.25$				$x = 1.5$			
Domain	Eif	$h^{\max}$	h^C_L	Sym	Eif	$h^{\max}$	h^C_L	Sym	Eif	$h^{\max}$	h^C_L	Sym
Blocksworld(400)	140	252	312	**339**	112	218	215	**295**	146	205	184	**284**
Nomystery(400)	239	333	381	**396**	183	264	207	**326**	177	225	93	**241**
Rovers(400)	248	305	**400**	398	259	303	**400**	387	268	315	**398**	352
TPP(400)	274	397	316	**400**	244	346	236	**400**	259	327	215	**400**
Sum (1600)	901	1287	1409	**1533**	798	1131	1058	**1407**	850	1072	890	**1276**

Table 1: Coverage (number of instances solved) on Eif20's benchmarks: (top) IPC benchmarks modified with a plan-cost bound, (bottom) benchmarks extended with action-set properties. Cost bounds set to x times the best known plan cost. Top left and bottom coverage for AllMUGS, top right coverage for cardinality-optimal OSP. "$h^{\max}$" ("h^C_L") is our explicit-search approach with $h^{\max}$ (h^C+nogood learning) pruning, "Sym" is our symbolic approach. "Eif"/"EifL" are Eif20's best-performing algorithms: (top) SysS without/with nogood learning, (bottom) SysW with trap learning. "Katz" is the state-of-the-art OSP solver. Best performance within each cost bound shown in **boldface**.

Nogood Learning We build on Eifler et al.'s [2020] adaptation of nogood learning to OSP. This algorithm takes as input a state s with remaining budget b_s, and a goal G which is not achievable from s within b_s, but $h^C(s, G) \leq b_s$. The algorithm returns an extended conjunction set $C' \supset C$ guaranteeing that $h^{C'}(s, G) > b_s$. Previous works use this algorithm to *refine* C (replace it with C') whenever a dead-end state s has been identified, from which the goal G cannot be reached. Each such update has the potential to generalize to states not seen so far, and may thus reduce future search effort.

Unlike previous works, our search here has no fixed goal G. Hence our approach differs in when and how C is refined. We run a depth-first search. This guarantees that, upon backtracking from a state s with remaining budget b_s, no $G_{new} \in \text{MUGS}(\mathcal{G}_{\text{curr}})$ can be achieved from s within b_s. When backtracking on s, we refine C so that s would be pruned according to (*): we find a set of conjunctions $C' \supset C$ so that $h^{C'}(s, G^{\text{hard}} \cup G_{new}) > b_s$ for all $G_{new} \in \text{MUGS}(\mathcal{G}_{\text{curr}})$. We do so by a separate call to Eifler et al.'s refinement method for each G_{new} where the inequality is not already satisfied.

5 Experiments with AllMUGS Algorithms

We implemented our AllMUGS algorithms in Fast Downward [Helmert, 2006], on top of Eif20's code base, using the SymBA [Torralba *et al.*, 2014] code for symbolic search. We use Eif20's benchmarks, comprising two parts. First, the classical-planning IPC benchmarks, modified by enforcing a

plan-cost bound. Second, four domains (resource-constrained planning by Nakhost et al. [2012], plus the Blocksworld) extended with G^{soft} encoding action-set properties, setting the original goals as G^{hard} and choosing the cost bound so that some but not all of G^{soft} can be satisfied. All experiments were run on Intel E5-2660 machines running at 2.20 GHz, with a time (memory) limit of 30min (4GB).

Table 1 shows coverage results. Our main interest here is AllMUGS (OSP is included as a side note, discussed below). Clearly, our explicit-state method with $h^{\max}$ (henceforth: **Exp**$h^{\mathbf{max}}$) and our symbolic method (**Sym**) both outclass Eif20's methods.[2] Consider first the IPC benchmarks (top left part of Table 1). Exp$h^{\max}$ increases overall coverage compared to the best previous method by 163 for $x = 0.25$, 203 for $x = 0.5$, and 165 for $x = 0.75$. For Sym, these numbers are 126, 152, and 149. Looking at individual domains, we see that Exp$h^{\max}$ and Sym are highly complementary. For (Eif, EifL, $h^{\max}$, h^C_L, Sym) the numbers of IPC domains with unique best coverage are (0, 1, 12, 1, 11) for $x = 0.25$, (0, 0, 16, 3, 12) for $x = 0.5$, and (0, 2, 17, 0, 18) for $x = 0.75$. So Eif20's methods stand out in just 3 cases in total, while Exp$h^{\max}$ and Sym stand out in 45 and 41 cases respectively.

On the benchmarks with AS properties (bottom table in Table 1), our new methods outclass Eif20's methods even more drastically. Interestingly, now Sym is in the lead by far. This is due to the smaller sets G^{soft} of properties to be analyzed. Sym is more effective at enumerating large state spaces, but suffers from the enumeration of goal subsets $G \subseteq G^{\text{soft}}$ in the meta-search (SysS/SysW). Explicit search with h^C (**Exp**h^C) suffers from the overhead of nogood learning on all G_{new}, cf. Section 4. Nevertheless, Exph^C stands out in some cases.

Consider finally, as a side remark, cardinality-optimal OSP (top right in Table 1). "Katz" here is the most recent OSP solver [Katz *et al.*, 2019]; Sym is extended to output a cardinality-optimal plan; Exp$h^{\max}$ data is repeated for convenience. Both our methods have higher overall coverage.

6 LTL Plan Properties

Eif20's AS properties lack expressivity in that they cannot capture temporal aspects of plans. Even simple properties like "P_0 should be delivered before P_1" cannot be expressed. Yet temporal properties clearly are natural and relevant for expressing user preferences, as e. g. in PDDL3 [Gerevini *et al.*, 2009]. LTL is a canonical language to address this.

We now show how LTL plan properties can be handled in Eif20's framework, with little overhead relative to AS properties. Section 6.1 outlines our compilation, Section 6.2 gives an example, Section 6.3 summarizes our experiments.

6.1 LTL$_f$ and Compilation into Goal Facts

There has been extensive prior work on the compilation of LTL plan preferences into PDDL [Edelkamp, 2006; Baier and McIlraith, 2006; Baier *et al.*, 2009; De Giacomo *et al.*, 2014; Torres and Baier, 2015]. Here we leverage this work, assembling previous methods in a way suiting our purposes.

[2]Eif/EifL results slightly differ from the ones previously reported by Eif20 because we fixed a bug in the implementation, but this does not significantly affect this comparison.

Plans are finite traces, in difference to the infinite traces assumed in LTL. We follow Baier and McIlraith [2006] in the solution to this problem. We adopt their LTL$_f$ language, a simple adaption of LTL interpreted over finite traces; and we use part of their compilation machinery. In LTL$_f$, the property "P_0 should be delivered before P_1" can, in the context of our NoMystery example from Figure 1, be defined as $\neg at(P_1, L_0)\ \mathcal{U}\ at(P_0, L_4)$. LTL$_f$ is interpreted over states, but when adding action effects identifying the last action applied, one can also talk about actions. The atoms A_i in Eif20's AS properties are then captured by $\Diamond \bigvee_{a \in A_i} a$.

Known compilations of LTL properties into PDDL proceed in two steps: (1) produce the property automaton, (2) compile that automaton into additional state variables and actions enforcing its execution alternatingly with the actions executed by the planner. We use Baier and McIlraith's [2006] tool for (1), producing an NFA. For (2), we cannot use Baier and McIlraith's tool as it relies on PDDL axioms which we do not support. We use (a re-implementation of) Edelkamp's [2006] tool instead. That tool takes as input an NFA and a PDDL instance, and outputs a modified instance with a goal fact g that is true at the end of a plan iff the NFA reached an accepting state. We implemented this on top of Eif20's framework, in the Fast Downward translator [Helmert, 2009] afer the grounding step. An important detail is that, as plan properties are soft goals in our context, we need to continue search paths even when one of the automata cannot reach an accepting state anymore. Edelkamp [2006] achieves this by additional actions in the compilation. Instead, we address this elegantly in step (1) by augmenting Baier and McIlraith's [2006] output NFA to be complete (a standard operation, adding a new accepting state reached by non-read input symbols).

6.2 An Example Analysis

To illustrate the kind of analysis afforded by the added capability to analyze LTL$_f$ properties, consider again our NoMystery example from Figure 1. Say we want to analyze the AS properties $1 : use(T_1, L_0, L_4)$, $2 : not\_use(T_0, L_0, L_1)$, and $3 : not\_use(T_1, L_1, L_2)$ in combination with the LTL$_f$ properties 4 : "load P_1 only once" and 5 : "deliver P_1 before P_2". Then we produce G^{soft} using the compilation for each property individually (we can even use the simpler AS-property compilation for 1–3, as the compilations are compatible). The MUGS then are $\{1, 3, 4\}, \{2, 3\}, \{3, 4, 5\}$. So, for example, the user question "Why is P_1 not delivered before P_2" can be answered with "Because then, either P_1 must be loaded more than once, or T_1 has to use the road between L_1 and L_2."

6.3 Experiments

We now evaluate the performance of LTL$_f$ plan-property dependency analysis. We examine scaling behavior over the number n of properties whose dependencies should be analyzed. This is relevant to gauge practical feasibility, as the worst-case number of MUGS is exponential in n. We use Eif20's benchmarks featuring AS properties. We focus on Sym, which scales best on these benchmarks (cf. the bottom part of Table 1); we briefly discuss Exph^C below.

Given the drastically improved performance relative to Eif20's algorithms, we had to create larger instances to make

Domain	Encoding	n (#properties) 1	2	3	4	5	6	7	8	9	10
AS vs LTL$_f$ encoding of AS properties											
Blocksworld	AS	10	7	6	5	5	3	3	3	3	2
	LTL$_f$	10	10	8	4	4	5	5	4	2	2
Nomystery	AS	9	8	7	7	8	5	6	7	6	6
	LTL$_f$	9	7	5	7	9	6	4	6	6	8
Rovers	AS	5	6	7	7	7	7	8	7	7	7
	LTL$_f$	7	5	3	5	6	6	6	6	7	7
TPP	AS	10	10	9	8	8	8	8	7	7	7
	LTL$_f$	10	10	10	10	9	9	8	8	7	8
LTL$_f$ properties not expressible as AS properties											
Blocksworld		10	10	7	2	3	1	2	0	0	0
Nomystery		4	5	4	2	1	0	0	0	0	0
Rovers		7	4	3	2	1	0	1	1	0	0
TPP		10	10	10	10	9	9	8	9	9	8

Table 2: Sym coverage as a function of the number n of plan properties whose dependencies must be analyzed ($x = 1.0$).

interesting observations about scaling in n. To this end, we increased the number of objects and hard goals up to a size region within which instances with large n become challenging for Sym. For each instance, we generated 10 AS properties and 10 LTL$_f$ properties not expressible as AS properties. Both were based on hand-made schemas instantiated randomly with concrete objects. The AS-property schemas are those by Eif20, and talk about use/disuse of road connections, transportation with the same vehicle, which object is used to achieve which goal, etc. The LTL$_f$-property schemas talk about the order in which goals are achieved, the frequency with which actions are used, and which facts are true together at some point along the plan.

Table 2 shows coverage results. Scaling over n is realized by fixing an arbitrary order of the 10 properties, and including them incrementally. We use $x = 1.0$ for the plan-cost bound; our results for other values of x are qualitatively similar.

Consider first the top part of Table 2, comparing LTL$_f$ encodings to AS encodings where both are possible. This serves to evaluate whether the more general LTL$_f$ methodology causes a computational overhead. In terms of coverage, the answer here is a clear "no". There is some variance in both directions, but nothing of a systematic nature. Figure 2 gives runtime data for a more fine-grained view. AS encodings tend to be faster more frequently. There is variance again though, and the difference is mostly moderate.

In the bottom part of Table 2, we see that scaling performance gets worse beyond AS properties. But the analysis is still reasonably effective for up to four or five plan properties, sometimes more. Of course, this depends also on instance size. Our intances here are slightly smaller than the ones by Nakhost et al. [2012], and are mid-range (ca. instance 7–13 out of 20) compared to the respective IPC test suites.

As before (bottom part of Table 1), Exph^C scales better than Sym in Rovers. It has coverage 9 or 10 for all values of n regardless of the property language.

7 Other Related Works

In constraint satisfaction problems, finding a minimum unsatisfiable core [Chinneck, 2007; Laborie, 2014] is related to finding a MUGS. Similar problems have been considered in constraint-based formulations of planning [Laborie and Ghallab, 1995; Yu *et al.*, 2017]. Problems involving some form of solvability borderline within a lattice of problem variants, like in Eif20's SysS and SysW, have a long tradition (e. g. [Reiter, 1987]). In particular, the diagnosis framework by Grastien et al. [2011; 2012] can cast maximum solvable goal subsets as preferred diagnoses. Perhaps this approach could profit from symbolic representations, as our work on AllMUGS here.

Eif20's framework that we follow up on here aims at plan-space explanation, which has previously been addressed in other forms for unsolvable tasks [Göbelbecker *et al.*, 2010; Sreedharan *et al.*, 2019]. It can also be viewed as domain/task analysis, or model checking applied to planning models, but in a form quite different from previous work (e. g. [Fox and Long, 1998; Rintanen, 2000; Vaquero *et al.*, 2013]).

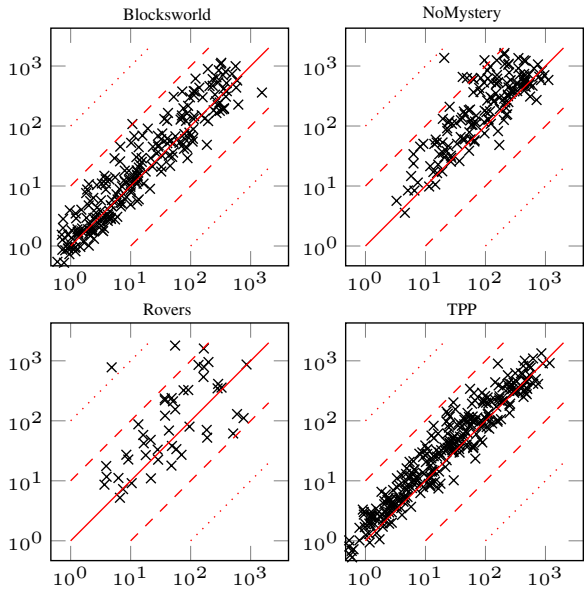

Figure 2: Sym runtime (s) on instances with AS properties, solved commonly for AS encoding (x-axis) vs. LTL$_f$ encoding (y-axis).

8 Conclusion

Eifler et al. [2020] introduced a framework approaching plan-space explanation as the identification of plan-property dependencies. We have extended their machinery to LTL plan properties, and introduced algorithms vastly improving computational performance. This provides a robust basis from which the framework can be further extended.

Future directions include temporal planning; answering deeper "why" questions (why does a plan-property entailment hold?); automatically identifying plan properties of interest in the given task/to a given user; and case studies in applications.

Acknowledgments

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-18-1-0245, and by the German Research Foundation (DFG) under grant 389792660 as part of TRR 248 (see https://perspicuous-computing.science).

References

[Baier and McIlraith, 2006] Jorge A. Baier and Sheila A. McIlraith. Planning with first-order temporally extended goals using heuristic search. In *Proc. AAAI*, pages 788–795, 2006.

[Baier *et al.*, 2009] Jorge A. Baier, Fahiem Bacchus, and Sheila A. McIlraith. A heuristic search approach to planning with temporally extended preferences. *AI*, 173(5-6):593–618, 2009.

[Cashmore *et al.*, 2019] Michael Cashmore, Anna Collins, Benjamin Krarup, Senka Krivic, Daniele Magazzeni, and David Smith. Towards explainable AI planning as a service. In *ICAPS XAIP*, 2019.

[Chinneck, 2007] John W. Chinneck. *Feasibility and Infeasibility in Optimization: Algorithms and Computational Methods*. Springer-Verlag, 2007.

[De Giacomo *et al.*, 2014] Giuseppe De Giacomo, Riccardo De Masellis, and Marco Montali. Reasoning on LTL on finite traces: Insensitivity to infiniteness. In *AAAI*, pages 1027–1033, 2014.

[Domshlak and Mirkis, 2015] Carmel Domshlak and Vitaly Mirkis. Deterministic oversubscription planning as heuristic search: Abstractions and reformulations. *JAIR*, 52:97–169, 2015.

[Edelkamp and Kissmann, 2009] Stefan Edelkamp and Peter Kissmann. Optimal symbolic planning with action costs and preferences. In *Proc. IJCAI*, pages 1690–1695, 2009.

[Edelkamp, 2006] Stefan Edelkamp. On the compilation of plan constraints and preferences. In *ICAPS*, pages 374–377, 2006.

[Eifler *et al.*, 2020] Rebecca Eifler, Michael Cashmore, Jörg Hoffmann, Daniele Magazzeni, and Marcel Steinmetz. A new approach to plan-space explanation: Analyzing plan-property dependencies in oversubscription planning. In *AAAI*, 2020.

[Fox and Long, 1998] Maria Fox and Derek Long. The automatic inference of state invariants in TIM. *JAIR*, 9:367–421, 1998.

[Fox *et al.*, 2017] Maria Fox, Derek Long, and Daniele Magazzeni. Explainable planning. In *ICAI XAI*, 2017.

[Gerevini *et al.*, 2009] Alfonso Gerevini, Patrik Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *AI*, 173(5-6):619–668, 2009.

[Göbelbecker *et al.*, 2010] Moritz Göbelbecker, Thomas Keller, Patrick Eyerich, Michael Brenner, and Bernhard Nebel. Coming up with good excuses: What to do when no plan can be found. In *Proc. ICAPS*, pages 81–88, 2010.

[Grastien *et al.*, 2011] Alban Grastien, Patrik Haslum, and Sylvie Thiébaux. Exhaustive diagnosis of discrete event systems through exploration of the hypothesis space. In *DX*, 2011.

[Grastien *et al.*, 2012] Alban Grastien, Patrik Haslum, and Sylvie Thiébaux. Conflict-based diagnosis of discrete event systems: Theory and practice. In *Proc. KR*, pages 489–499, 2012.

[Haslum and Geffner, 2000] Patrik Haslum and Hector Geffner. Admissible heuristics for optimal planning. In *Proc. AIPS*, pages 140–149, 2000.

[Haslum, 2012] Patrik Haslum. Incremental lower bounds for additive cost planning problems. In *Proc. ICAPS*, pages 74–82, 2012.

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

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

[Katz *et al.*, 2019] Michael Katz, Emil Keyder, Dominik Winterer, and Florian Pommerening. Oversubscription planning as classical planning with multiple cost functions. In *ICAPS*, pages 237–245, 2019.

[Krarup *et al.*, 2019] Benjamin Krarup, Michael Cashmore, Daniele Magazzeni, and Tim Miller. Towards model-based contrastive explanations for explainable planning. In *ICAPS XAIP*, 2019.

[Laborie and Ghallab, 1995] Philippe Laborie and Malik Ghallab. Planning with sharable resource constraints. In *IJCAI*, pages 1643–1651, 1995.

[Laborie, 2014] Philippe Laborie. An optimal iterative algorithm for extracting MUCs in a black-box constraint network. In *Proc. ECAI*, pages 1051–1052, 2014.

[McMillan, 1993] Kenneth L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

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

[Nakhost *et al.*, 2012] Hootan Nakhost, Jörg Hoffmann, and Martin Müller. Resource-constrained planning: A Monte Carlo random walk approach. In *Proc. ICAPS*, pages 181–189, 2012.

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

[Rintanen, 2000] Jussi Rintanen. An iterative algorithm for synthesizing invariants. In *Proc. AAAI*, pages 806–811, 2000.

[Smith, 2004] David E. Smith. Choosing objectives in oversubscription planning. In *ICAPS*, pages 393–401, 2004.

[Smith, 2012] David Smith. Planning as an iterative process. In *AAAI*, pages 2180–2185, 2012.

[Sreedharan *et al.*, 2019] Sarath Sreedharan, Siddharth Srivastava, David Smith, and Subbarao Kambhampati. Why can't you do that HAL? explaining unsolvability of planning tasks. In *IJCAI*, pages 1422–1430, 2019.

[Steinmetz and Hoffmann, 2017a] Marcel Steinmetz and Jörg Hoffmann. Search and learn: On dead-end detectors, the traps they set, and trap learning. In *IJCAI*, pages 4398–4404, 2017.

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

[Torralba *et al.*, 2014] Álvaro Torralba, Vidal Alcázar, Daniel Borrajo, Peter Kissmann, and Stefan Edelkamp. SymBA*: A symbolic bidirectional A* planner. In *IPC 2014 abstracts*, 2014.

[Torralba *et al.*, 2017] Álvaro Torralba, Vidal Alcázar, Peter Kissmann, and Stefan Edelkamp. Efficient symbolic search for cost-optimal planning. *AI*, 242:52–79, 2017.

[Torres and Baier, 2015] Jorge Torres and Jorge A Baier. Polynomial-time reformulations of LTL temporally extended goals into final-state goals. In *IJCAI*, pages 1696–1703, 2015.

[Vaquero *et al.*, 2013] Tiago Stegun Vaquero, José Reinaldo Silva, Flavio Tonidandel, and J. Christopher Beck. itSIMPLE: Towards an integrated design system for real planning applications. *Knowledge Engineering Review*, 28(2):215–230, 2013.

[Yu *et al.*, 2017] Peng Yu, Brian Charles Williams, Cheng Fang, Jing Cui, and Patrik Haslum. Resolving over-constrained temporal problems with uncertainty through conflict-directed relaxation. *JAIR*, 60:425–490, 2017.