Value Driven Landmarks for Oversubscription Planning

Daniel Muller, Erez Karpas
Technion - Israel Institute of Technology
Haifa, Israel

Abstract

Oversubscription planning is the problem of choosing an action sequence which reaches a state with a high utility, given a budget for total action cost. Most previous work on oversubscription planning was restricted to only non-negative utility functions and 0-binary utility functions. While this restriction allows using techniques similar to partial satisfaction planning, it limits the expressivity of the formalism. In this paper, we address oversubscription planning with general additive utility functions over a finite-domain representation. We introduce the notions of net utility of an action, and of a gross positive action. Using these notions, we prove several properties about the structure of an optimal plan, which are then compiled into a classical planning problem. The landmarks of this classical planning problem are value driven landmarks of the original oversubscription problem, that is, they must occur in any action sequence which improves utility. An empirical evaluation demonstrates that these landmarks are more informative than previous state-of-the-art methods for landmark discovery for oversubscription planning, and lead to better planning performance.

Introduction

Oversubscription planning (Smith 2004), hereinafter referred to as OSP, deals with achieving a state with a high utility value, given a budget on total action cost. This formulation allows us to handle situations with under-constrained resources, which do not allow us to achieve all possible goal facts. This is in contrast to classical planning, in which the objective is to find a cheap plan which achieves *all* goal facts. In optimal OSP and optimal classical planning, the tasks are further constrained to finding a path which achieves a state with maximal utility, and, to finding the cheapest cost path which achieves the goal, respectively.

Over the years, the theory and practice of classical planning have been studied and advanced much more intensively compared to OSP. Recent work (Mirkis and Domshlak 2013; 2014; Domshlak and Mirkis 2015) made several contributions aiming at improving the scalability of OSP solvers. In particular, they developed a planner which exploits standard landmark discovery tools of classical planning, as well as abstractions for solving OSP problems. In this paper, we propose a technique which allows us to discover more informative landmarks than previous methods. Furthermore, our technique is applicable for general additive utility OSP tasks. More specifically, we introduce the notion of the *net utility* of an action, which is the net change in utility after applying this action. We prove that any optimal plan must end with an action which has positive net utility. We also introduce the notion of a *gross positive* action, which improves the utility for at least one of the state variables, although possibly at the cost of a lower utility for other variables. We then combine these and prove that any optimal plan must contain a gross positive action (which might have negative net utility), which achieves some fact that is maintained until the end of the plan, which must be an action with positive net utility. This approach of *improving* differs from previous approaches which attempted to *collect* valuable facts and allow us to handle general additive utility functions, rather than only non-negative utilities.

To leverage these properties, we introduce a compilation of an OSP task to classical planning, which forces a plan to conform to these properties. The landmarks of this classical planning task constitute value driven landmarks (called ε-landmarks by Domshlak and Mirkis (2015)), that is, landmarks which must occur in every plan that improves over the utility of the initial state. We also introduce a slight modification to the inc-compile-and-BFBB planner (Domshlak and Mirkis 2015), which leverages our definitions of net positive action and gross positive actions and our observations on these definitions. An empirical evaluation shows that the landmarks we discover are more informative than those of Domshlak and Mirkis (2015), and that our modified planner outperforms state-of-the-art OSP planners.

Preliminaries

Some auxiliary notation first. For $k \in \mathbb{N}^+$, by $[k]$ we denote the set $\{1, 2, \ldots, k\}$. An assignment of a variable v to value d is denoted by $\langle v/d \rangle$ and referred as a *proposition* or *fact*. For a partial assignment p to V, let $\mathcal{V}(p) \subseteq V$ denote the subset of variables instantiated by p, and, for $v \in \mathcal{V}(p)$, $p[v]$ denote the value provided by p to the variable v.

Planning Framework. In what follows we adopt the OSP model representation of Domshlak and Mirkis (2015), adopting a language close to the SAS$^+$ language for classical planning (Bäckström and Klein 1991; Bäckström and Nebel 1995). A deterministic **oversubscription planning (OSP)** task is given by a sextuple $\Pi = \langle V, s_0, u; O, c, b\rangle$, where $V = \{v_1, \dots, v_n\}$ is a finite set of finite-domain *state variables*, with each complete assignment to V representing a *state*, and $S = dom(v_1) \times \cdots \times dom(v_n)$ being the *state space* of the task; $s_0 \in S$ is a designated *initial state*; u is an efficiently computable *state utility* function $u : S \to \mathbb{R}$; O is a finite set of *actions*, with each action $o \in O$ being represented by a pair $\langle \mathsf{pre}(o), \mathsf{eff}(o)\rangle$ of partial assignments to V, called *preconditions* and *effects* of o, respectively; $c : O \to \mathbb{R}^{0+}$ is an *action cost* function; $b \in \mathbb{R}^{0+}$ is a *cost budget* allowed for the task. Action o is applicable in a state s if $s[v] = \mathsf{pre}(o)[v]$ for all $v \in \mathcal{V}(\mathsf{pre}(o))$. Applying o changes the value of each $v \in \mathcal{V}(\mathsf{eff}(o))$ to $\mathsf{eff}(o)[v]$, and the resulting state is denoted by $s[\![o]\!]$. Sequential application of actions $\langle o_1, \dots, o_m\rangle$ denoted by π, called a plan for s if it is applicable in s and $c(\pi) \leq b$. We assume a *general additive utility* function with *multi-valued variables*, defined as $u(s) = \sum_{\langle v/d\rangle \in s} u_v(d)$, with $u_v(d) \in \mathbb{R}$ for all variable-value pairs $\langle v/d\rangle$. That is, negative utility values are allowed and each variable can get many value assignment with different utility available for each variable-value pair.

A deterministic **classical planning** task, is given by $\Pi = \langle V, s_0, G; O, c\rangle$, where the utility function and cost budget are replaced with a hard goal constraint which is a partial assignment on state variables $G \subseteq V$.

Disjunctive Action Landmarks. Fact landmark are propositions that must be true at some point in every solution plan for a given planning task (Hoffmann, Porteous, and Sebastia 2004). Similarly to landmarks over facts, **disjunctive action landmarks** (Vidal and Geffner 2006; Zhu and Givan 2004), correspond to a set of actions such that every plan contains at least one action from that set. While landmarks are widely used in (both satisficing and optimal) classical planning (Karpas and Domshlak 2009; Helmert and Domshlak 2009; Coles and Coles 2011; Domshlak, Katz, and Lefler 2012; Bonet and Helmert 2010; Pommerening and Helmert 2013), the first framework for their exploitation in the context of OSP has been introduced by Domshlak and Mirkis (2015) only recently. This framework basic construct is the notion of **ε-landmarks**.

An **ε-landmark** for a state s is a set of actions that is applied in every plan π that achieves *something valuable*, that is, by every plan having a positive value. For instance, with the disjunctive action landmarks we use here, if $L \subseteq O$ is an ε-landmark for s, then every plan having a positive value end-state contains an action from L.

ε-landmarks can be used to construct a **budget reduction compilation** (Domshlak and Mirkis 2015), as follows:

- Given an OSP task Π, we generate the ε-compilation Π_ε of Π, where Π_ε is a classical planning task that simply extends the structure of Π with a set of zero-cost actions such that applying any of them corresponds to verifying that a positive value can be achieved in Π.
- Using classical planning tools, a set of landmarks $\mathcal{L}$ is generated for Π_ε, along with an admissible landmark cost function $lcost : \mathcal{L} \to \mathbb{R}^{0+}$ where $\sum_{L\in\mathcal{L}} lcost(L) \leq h^*(s)$ (Katz and Domshlak 2010; Karpas and Domshlak 2009; Helmert and Domshlak 2009).
- Performing budget reducing compilation by compiling $(\mathcal{L}, lcost)$ into Π, obtaining an OSP task $\Pi_\mathcal{L}$, where $\Pi_\mathcal{L}$ extends the structure of Π by mirroring the actions of each ε-landmark L_i with their "cheaper by $lcost(L_i)$" versions. To compensate for the discounted actions, budget b is reduced by $\sum_{i=1}^{n} lcost(L_i)$. Domshlak and Mirkis (2015) devised two versions of polynomial and sound budget reducing compilation; A basic compilation $\Pi_\mathcal{L}$ applied on a pairwise disjoint landmarks and extended with auxiliary control structure, a *generalized budget reducing compilation* applied on an arbitrary (non-disjoint) sets of ε-landmarks. This compilation can then be used within a heuristic search algorithm, as described next.

Heuristic Search for OSP. Best-First-Branch-and-Bound (*BFBB*) for OSP must rely on admissible utility-upper-bounding heuristic function (within budget b restrictions) $h : S \times \mathbb{R}^{0+} \to \mathbb{R}^{0+}$ to estimate the true utility $h^*(s, b)$.

BFBB maintains a queue in decreasing order of $h(s\langle n\rangle, b - g(n))$ and the best solution n^* found so far. All generated nodes n evaluated no higher than $u(n^*)$ and nodes with cost-so-far $g(n)$ higher than the problem's budget b are pruned. When the node-list becomes empty, or the node selected from the list promises less than the lower bound, the plan associated with the best solution n^* is returned. If the heuristic h is admissible, then the returned plan is guaranteed to be optimal.

inc-compile-and-BFBB (Domshlak and Mirkis 2015) is an incremental version of *BFBB*. Additionally to the best solution so far n^*, inc-compile-and-BFBB maintains a set of reference states S_{ref}, which is updated with any new, best-do-far state which has reached n^*. For each $s_i \in S_{\text{ref}}$ and each valuable fact $g \in V \setminus s_i$ a zero cost action is added to O which indicates for a valuable fact that is not in s_i. Applying all these zero cost actions verifies an achievement of subset of *good* facts that included in no state from S_{ref}. With each update of S_{ref}, *BFBB* is restarted with a new set of ε-landmarks created on the basis of the *current* S_{ref}, that is derived concerning achieving something that has not yet been seen so far. These new, more informed *ε-landmarks* are compiled back into the original task for next iteration.

The Properties of Optimal Plans

In this section, we present several properties of the structure of optimal plans. In the following section, we present a full compilation and a solver exploiting these properties. We will refer to a state that improves over the utility of the initial state as a *valuable state*, and to a sequence of actions reaching such a state as a *valuable plan*.

The Utility of Actions

Considering negative utility values in OSP, because of the dependencies between propositions, it is possible that collecting positive utility values might require collecting negative utility values on the way. In particular, it is possible that the same action achieving positive value propositions also achieves some negative value propositions. For an OSP action o, the *total outcome utility* $u^{out}(o) = \sum_{v \in \mathcal{V}(\mathsf{eff}(o))} u_v(\mathsf{eff}(o)[v])$, capture these inter-state variables dependencies in a wider context.

Negative interactions are not only confined to problems with variables of negative utility. Even when variables take a non-negative form, an action achieving valuable propositions might lose more valuable propositions that have been held at the origin state. The *net utility* of an action definition provides a wider context of achievement evaluation, by taking in account both target state and its origin, and captures the intra-state dependencies, in addition to the inter-state dependencies (which captured with the total outcome utility). For clarity of presentation, we present the net utility of an action in SAS representation, under the assumption that for every effect variable a precondition is defined, so the notation is well defined. Later, we relax our simplifying assumption with the *selective action split*. The net utility of an action o is defined as follows:

Definition 1. *For an OSP planning task action o, the* **net utility** *of o is* $u(o) = \sum_{v \in \mathcal{V}(\mathsf{eff}(o))}[u(\mathsf{eff}(o)[v]) - u(\mathsf{pre}(o)[v])]$.

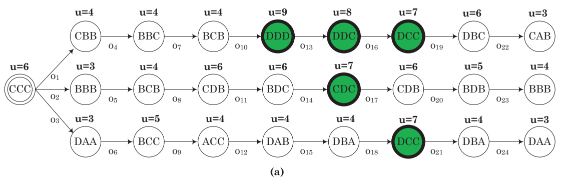
The definition of net utility will help us to derive more informative ε-landmarks, and further reduce the search space.

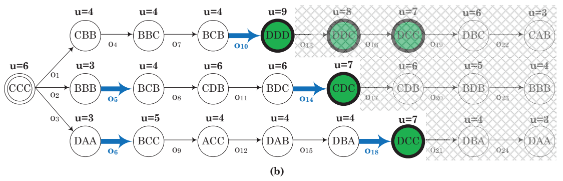
High Level Overview

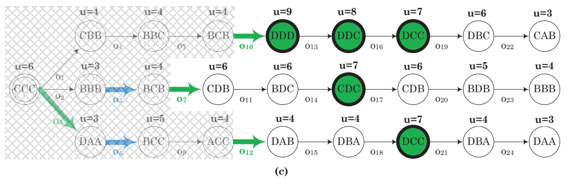
In what follows, we present an interrelated set of properties of optimal plans, which allow us to define a pattern of valuable plans in OSP tasks. Our landmark generation method exploits these optimal plan properties on the level of actions and sequence of actions, in contrast to the traditional "collecting" approach which generates landmarks to valuable facts, which based on properties of the effect list of a single action. These properties are used to generate refined and lengthier landmarks which are the input for Mirkis and Domshlak budget reduction procedure and improve the pruning mechanism which is based on these landmarks. A running example in Figure 1 shows all plans for a given OSP task and will be referred to demonstrate these properties. In this OSP task an additive utility function is defined over three state variables with the same domain $dom = \{A, B, C, D\}$, by $u(A) = 0, u(B) = 1, u(C) = 2$ and $u(D) = 3$ for each variable.

In Lemma 1, we prove that any valuable plan terminates with a net positive utility action. The colored in blue arrows illustrated in Figure 1(b) indicate such net positive utility action candidates to finish the search with an optimal state in hand. The green colored actions depicted in Figure 1(c) are gross positive actions, whose effects improve over at least one of the variables in the initial state, and may have total negative utility by achieving worse utility for some other variables. We prove in Lemma 2 that such actions must occur in any optimal plan, prior to a net positive action application which terminates the search.

A special case is an action that is both gross positive and has positive net utility, for example, action o_{10} in Figure 1(d). In a strictly positive utility setting with an 0-binary utility function and an initial state with utility zero, the net positive actions are also gross positive. However, in general







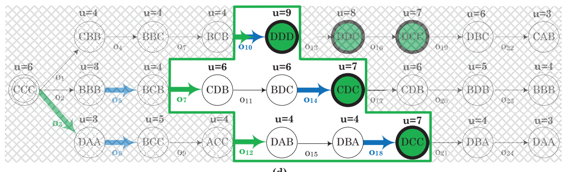


Figure 1: An example of an OSP task, with (a) illustrating a search space with valuable states depicted in green. The grayed out zone in (b) shows the pruning from Lemma 1 with net positive actions depicted with thick blue arrows. The region pruned by exploiting gross positive actions (Lemma 2) is grayed out in (c). Finally (d) shows the region that is pruned by combining both of the above.

additive utility settings, these actions are likely to be distinct. For example, the net positive utility actions o_5 and o_6 in our running example lead to states with lower utility compared to the initial state. A maintained achievement property, which we prove in Lemma 3 complements Lemma 2 and ensures that an achieved effect of a gross positive action remains in hand when net positive actions are examined as candidates to terminate the search.

Finally, in Theorem 4, we combine all these claims and define a so-called 'window of opportunity' to terminate a plan with a valuable state in hand, as illustrated in Figure 1(d). We construct a classical planning task, which encodes the above mentioned constraints on the 'window of opportunity'. The landmarks for this classical planning task, are *value landmarks* which can guide us towards a valuable state in the original OSP task.

Net Positive Actions

At first glance, the net utility value of an action does not have a clear role in our reasoning; It is very much possible that optimal plans will contain many actions with negative net utility since it allows for achieving valuable facts that outbalance the respective loss. However, Lemma 1 shows that actions with positive net value do play a characteristic role in optimal plans. The proof of Lemma 1 is rather simple and, in plain words, the lemma states that, for each plan π, there is a plan π' that; (i) ends with a positive net utility action, (ii) is at most as costly as π, and (iii) is at least as valuable as π.

Lemma 1. *Given an OSP task Π with a general additive utility function u, for any plan π for Π such that $u(s[\![\pi]\!]) > u(s_0)$, there is a prefix π' of π such that: 1. $u(s_0[\![\pi]\!]) \leq u(s_0[\![\pi']\!])$, and 2. for the last action o_{last} along π', we have $u(o_{last}) > 0$.*

Proof. The proof is by induction on the plan length n. For $n = 1$, we have $\pi = \langle o_1 \rangle$, and since $u(s[\![\pi]\!]) > u(s_0)$, the operator o_1 has a positive net utility value. Hence, $\pi' = \pi$ satisfies the claims. Assuming that the claim holds for $n \geq 1$, we now prove it for $n + 1$.

Considering a plan $\pi = \langle o_1, \ldots, o_{n+1} \rangle$, for $k \in [n+1]$, let π_i denote the prefix of π consisting of its first i operators. If the last operator o_{n+1} along π has a positive net utility value, then we are done with $\pi' = \pi$. Otherwise, if o_{n+1} has either negative or zero net utility value, then $u(s[\![\pi]\!]) > u(s_0)$ in particular implies $u(s[\![\pi_n]\!]) > u(s_0)$. If π' is a prefix of π_n that satisfies the lemma by our assumption of induction, then π' also satisfies the lemma with respect to π since $u(\pi') \geq u(s[\![\pi_n]\!]) \geq u(s[\![\pi]\!])$. □

Relying on Lemma 1, we can now extract ε-landmarks which focus on actions with positive net value as candidates to end the search process. Specifically, we can devise a polynomial time compilation to classical planning, that extends the structure of Π with a goal variable g, and for any *net positive action o*, the effects are extended with the proposition $\langle g/1 \rangle$. Applying any of those actions indicates a positive value improvement in Π. With this compilation we can use any method for classical planning landmark extraction to derive more informative *ε-landmarks* for Π, due to a refinement of goal-achieving actions.

Gross Positive Actions

Considering tasks with a valuable initial state s_0, a net positive utility action might be applied during the search, leading to a valuable state s with lower utility $u(s) < u(s_0)$. Since landmarks hold for any valuable plan, they must hold for a plan reaching state s.

We are no longer interested in a search of plans that "achieve something", but in search for plans that achieve an *improvement* over the initial state s_0.

Definition 2. *Given an OSP task $\Pi = \langle V, s_0, u; O, c, b \rangle$ and a state s, let* **improved(s)** *be the set of all propositions $\langle v/d \rangle \in \mathcal{D}$ such that $u_v(d) > u_v(s[v])$.*

Based on the definition of **improved(s)** we define a *gross positive action*, which achieves an improvement over at least one variable from the initial state.

Definition 3. *Given an OSP task $\Pi = \langle V, s_0, u; O, c, b \rangle$ and a state s, the* **gross positive actions** *relative to s $O_{gPos}(s) \subseteq O$ of Π are $O_{gPos}(s) = \{o \in O, \mathsf{eff}(o) \cap \mathsf{improved}(s) \neq \emptyset\}$.*

For now, we will use the initial state s_0 as the reference state to improve over. Lemma 2 is based on the definition of the *gross positive actions* and shows that gross positive actions play a characteristic role in optimal plans. The proof of Lemma 2, in plain words, claims that, along any valuable plan π for an OSP task Π that improves over the given state, there is at least one *gross positive action*, which improves the utility for at least one of the state variables (although possibly at the cost of a lower utility for other variables).

Lemma 2. *Given an OSP task $\Pi = \langle V, s_0, u; O, c, b \rangle$, and a state s, for any plan π that improves over s, that is, $u(s_0[\![\pi]\!]) > u(s)$, it holds that: 1. $s_0[\![\pi]\!] \cap \mathsf{improved}(s) \neq \emptyset$, and 2. $\pi \cap O_{gPos}(s) \neq \emptyset$, that is, at least one action along π is gross positive.*

Proof. Assuming to the contrary that $s_0[\![\pi]\!] \cap \mathsf{improved}(s) = \emptyset$, by definition of $\mathsf{improved}(s)$ we have $u_v(s[\![\pi]\!][v]) \leq u_v(s[v])$, and thus $\sum_{\langle v/d \rangle \in s[\![\pi]\!]} u_v(d) \leq \sum_{\langle v/d \rangle \in s_0} u_v(d)$ and, in turn, $u(s[\![\pi]\!]) \leq u(s)$, contradicting the assumption of the lemma that π improves over s.

Since $s[\![\pi]\!] \cap \mathsf{improved}(s) \neq \emptyset$, consider any proposition $\langle v/d \rangle \in s[\![\pi]\!] \cap \mathsf{improved}(s)$. By definition of $\mathsf{improved}(s)$, $\langle v/d \rangle \notin s$, π must contain an operator o achieving $\langle v/d \rangle$, and by definition of $O_{gPos}(s)$, $o \in O_{gPos}(s)$. □

Applying any of those actions indicates a valuable achievement in Π. As illustrated in Figure 1(c) in our running example, the number of the landmarks derived with focus on gross positive utility actions can be substantially increased.

The Window of Opportunity to Improve

In tasks with an initial state carrying the lowest possible utility, always, all the constraints are satisfied by the last net positive action application, which makes Lemma 1 a sufficient condition. When it comes to tasks with an initial state utility $u(s_0)$ greater than the lowest possible, this set of separate events is likely to happen. When the *gross positive action* and the *net positive action* are different events, achievements may be lost between the occurrences of both of them.

Furthermore, Lemma 1 is stated concerning the net utility of an action, which is a relative term. The decision if the net utility of an action is enough to improve the initial state, depends on which state the action is applied in. When the initial state is the lowest possible, obviously, every net positive utility action is a good indicator for improvement. On the other hand, gross positive actions are defined in absolute terms, and allow for evaluation of achievements in relation to a reference state, in the context of a sequence of actions rather than a single action. The integration of those two properties to criteria for optimal plans with any initial utility state is through timing. The occurrence of a gross positive action establishes a dynamic reference point. Once a reference point is established, a net positive utility improvement must occur in relation to that point. The length of the 'window of opportunity' to achieve an improvement over the initial state is defined by the time that the achievement of the gross positive action keeps holding, that is, an improved proposition (or a set of propositions) over the initial state from $\mathsf{improved}(s_0)$ holds. When there is no achievement yet, or the achievement is not maintained, the net positive utility actions (which are not gross positive) do not bring a sustainable (global) benefit. To acquire more accurate landmarks, we have to capture the information about the occurrence of an effective gross-positive achievement, which allows for recognition of the 'zone' in which an effective search can terminate. A net positive action that is applied out of that 'zone' can be safely treated as just an intermediate action along with a plan.

Maintained Achievements. In Lemma 3 we define the maintained proposition property of sequential application of actions in a deterministic scenario. This notation is similar to the *maintenance goals* notation, which is used in the context of temporal planing (Haddawy and Hanks 1993; 1998). The proof of Lemma 3 is rather simple, it finalizes the claim in Lemma 2 by defining the 'goal area' where the plan may terminate with achievement in hand. In plain words, Lemma 3 states that for each improving plan π over the initial state there is a suffix plan π' such that (i) starts with a gross positive action o_{grs}, and (ii) a valuable proposition achieved by o_{grs} is maintained along $\pi^{'}$.

Lemma 3. *Given an OSP task* $\Pi = \langle V, s_0, u; O, c, b\rangle$, *and a state* s, *for any plan* π *that improves over* s, *that is,* $u(s_0[\![\pi]\!]) > u(s)$, *there is a suffix* π' *of* $\pi = \pi_0 \cdot \pi'$ *such that: 1.* π' *starts with a gross positive action* o_{grs}, *and 2. for some* $\langle v/d\rangle \in \mathsf{eff}(o_{grs}) \cap \mathsf{improved}(s)$, $\langle v/d\rangle$ *is maintained along* π', *that is, for each non-empty prefix* π'' *of* π', $\langle v/d\rangle \in s_0[\![\pi_0]\!][\![\pi'']\!]$.

Proof. Let $\pi = \langle o_1, \ldots, o_n\rangle$ be a plan for Π such that $u(s_0[\![\pi]\!]) > u(s)$. By Lemma 2, we have $s_0[\![\pi]\!] \cap \mathsf{improved}(s) \neq \emptyset$, and for each $\langle v/d\rangle \in s_0[\![\pi]\!] \cap \mathsf{improved}(s)$, the plan π contains an operator achieving $\langle v/d\rangle$. Consider any such proposition $\langle v/d\rangle \in s_0[\![\pi]\!] \cap \mathsf{improved}(s)$, and let o_i, $i \in [n]$, be the last instance of an operator along π that achieves $\langle v/d\rangle$. Since $\langle v/d\rangle \in s_0[\![\pi]\!]$, then no operator instance along the suffix $\pi' = \langle o_i, \ldots, o_n\rangle$ of π changes the value of v, and thus π' satisfies the claim of the lemma. □

Synergistic Criteria for a Valuable Plan. Given an OSP task Π with a general additive utility function and an initial state s_0, considering a plan π for Π with $u(s_0[\![\pi]\!]) > u(s_0)$, from Lemma 1, Lemma 2 and Lemma 3 we know the following properties for a valuable plan π: (1): for the last action o along π, we have $u(o) > 0$, (2): π contains at least one gross positive action, and (3): an achievement of at least one gross positive action is maintained, that is, holds along a suffix of π. Putting it all together, we are now able to devise additional constraints on valuable plans for OSP tasks. Any valuable plan π has a valuable partial plan.

Consider a plan $\pi = \langle o_1, \ldots, o_i, \ldots o_n\rangle$, with $i \in [n]$ where o_i is a *gross positive action*, that achieves a valuable proposition $\langle v/d\rangle \in \mathsf{improved}(s_0)$, yet with total state utility $u(s_0[\![\langle o_1, \ldots, o_i\rangle]\!]) < u(s_0)$. Obviously, the sequence of actions $\langle o_{i+1}, \ldots, o_n\rangle$ must include a net positive action. In this case the rest of the plan $\pi' = \langle o_i, \ldots o_n\rangle$ is dedicated to damage reduction over the variables from $V \setminus v$ that obtained on the way to collect the valuable proposition $\langle v/d\rangle \in \mathsf{improved}(s_0)$. In this case, there is an *ordering constraint* over the occurrence of applying a *gross positive action* and *net positive action*, where o_i must occur at some point before applying the net positive action o_n. Furthermore, along the execution of the partial plan $\pi' = \langle o_i, \ldots o_n\rangle$ the proposition $\langle v/d\rangle$ must be maintained by any action o_j, where $j \in [i, n]$.

The maintained proposition property proved in Lemma 3 allows us to combine into Theorem 4, through an *ordering constraint*, the properties provided in Lemma 1 and Lemma 2. Theorem 4 states that, for each valuable plan π, there is a *partial valuable plan* $\pi^{''}$, which is "window of opportunity" to terminate the search with an optimal end- state.

Theorem 4. *Given an OSP task* Π *with an additive utility function* u, *and a state* s *with* $u(s) \geq u(s_0)$, *for any plan* π *for* Π *with* $u(s_0[\![\pi]\!]) > u(s)$, *there is a suffix* π' *of* π, *such that:*

1. *The last action* o_l *of* π' *is a net positive action, and*
2. *The first action* o_f *of* π' *is a gross positive action relative to* s, *i.e.,* $o_f \in o_{grs}(s)$, *and*
3. *There exists some effect* $\langle v/d\rangle$ *of* o_f, *which is maintained throughout* π'.

Proof. The proof is immediate from the combination of Lemmas 1, 2, and 3. □

Relying on Theorem 4, we can define a more informative version of the compilation. Specifically, the compilation is extended with an auxiliary control structure of zero cost actions and zero utility propositions that restrict plans to terminate in the "window of opportunity" for improvement.

Multiple Action Repetitions. Many OSP domains allow for repetition of an action during a plan. In particular, an optimal plan π for an OSP task Π may involve multiple occurrences of *net positive actions* and *gross positive actions*. Considering an optimal plan $\pi = \langle o_1, \ldots, o_i, \ldots o_n\rangle$, with $k \in [n]$, Lemma 2 shows that a plan π that achieves an *improvement* over the initial state s_0 has to contain at least one *gross positive action*. The valuable proposition that is achieved with this gross positive action holds at the end-state of π. This necessary condition for optimal planning is beneficial for plans with no repeated actions. When multiple action repetition occurs, this condition is weaker and can lead to false recognition of sub plans as valuable plans.

Consider a plan $\pi = \langle o_1, \ldots, o_i, \ldots, o_j, \ldots o_n\rangle$, with $i, j \in [n]$, such that $i < j \leq n$ and a valuable proposition $\langle v/d\rangle \in \mathsf{eff}(o_i)$, such that $\langle v/d\rangle \in \mathsf{improved}(s_0)$, let the end-state be $s = s_0[\![\pi]\!]$ such that $u(s) > u(s_0)$. Suppose that, an action o_j such that $\langle v/d\rangle \in \mathsf{pre}(o_j)$ and $\{\langle v''/d''\rangle, \langle v/d\rangle\} \in \mathsf{eff}(o_j)$, where $d \neq d', v \neq v'$ and $\langle v''/d''\rangle \in \mathsf{pre}(o_n)$. In this example, the proposition $\langle v/d\rangle \in \mathsf{improved}(s_0)$ is achieved by an intermediate action o_i to satisfy a precondition of action o_j and lost immediately after, i.e. not maintained. Obviously, the proposition $\langle v/d\rangle \in s \cap \mathsf{improved}(s_0)$ must be re-acquired again, and the plan may include another occurrence of this action, that is, $\pi = \langle o_1, \ldots, o_i, \ldots, o_j, \ldots o_i, \ldots o_n\rangle$ so that $\langle v/d\rangle$ will hold at the end-state, that is, $\langle v/d\rangle \in s \cap \mathsf{improved}(s_0) \subseteq s$. Hence, multiple occurrence of gross positive action o_j in the action sequence $\pi = \langle o_i, \dots, o_j, \dots o_i\rangle$, may miss landmarks for this sequence.

To acquire more accurate landmarks we have to capture this information in the compiled classical planning task Π_ε. We allow for multiple action repetition by duplicating gross positive actions and extending each with an auxiliary control indicator g_{grs} as follows; The action o_{nm} is associated with a proposition $\langle v/d\rangle$ that is not-maintained, and constructed as $\mathsf{pre}(o_{nm}) = \mathsf{pre}(o) \cup \langle g_{grs}/0\rangle$, $\mathsf{eff}(o_{nm}) = \mathsf{eff}(o)$, $c(o_{nm}) = c(o)$. The action o_m is associated with a maintained proposition $\langle v/d\rangle$, o_{nm} can be applied only once along plan π and constructed as $\mathsf{pre}(o_m) = \mathsf{pre}(o) \cup \langle g_{grs}/0\rangle$, $\mathsf{eff}(o_m) = \mathsf{eff}(o) \cup \langle g_{grs}/1\rangle$, $c(o_m) = c(o)$. Since o is split to o_{nm} and o_m, it may occur in different landmarks.

The Valuable Plan Compilation

Relying on Theorem 4, In what follows, we focus on the **value driven landmarks** for an OSP task Π.

Definition 4. *Let* $\Pi = \langle V, s_0, u; O, c, b\rangle$ *be an OSP task and* $\mathsf{improved}(s)$*(with* s_0 *at the start) be the set of all propositions* $\langle v/d\rangle \in \mathcal{D}$ *such that* $u_v(d) > u_v(s[v])$. *The* **valuable plan compilation** *of* Π *is a classical planning task* $\Pi_\uparrow = \langle V_\uparrow, s_{0\uparrow}, G_\uparrow; O_\uparrow, c_\uparrow\rangle$ *constructed as* $V_\uparrow = V \cup E \cup Y$ *where* $E = \{srch, get, ver, end, g_{grs}, g_{net}, g\}$ *with* $dom = \{1, 0\}$ *and,* $Y = \{y^v_d \mid v \in V, d \in dom(v)\}$, $s_{0\uparrow} = s_0 \cup \{\langle y^v_d/0\rangle \mid y^v_d \in Y\} \cup \{\langle srch/1\rangle, \langle get/0\rangle, \langle g_{grs}/0\rangle, \langle g_{net}/0\rangle, \langle end/0\rangle, \langle g/0\rangle\}$, $G_\uparrow = \{\langle g/1\rangle\}$, *and* $O_\uparrow$ *constructed of sets of actions as follows:*

- $\{o_\mathsf{g} \mid o \in O, u(o) > 0, \mathsf{eff}(o) \cap \mathsf{improved}(s) \neq \emptyset\}$ *where*
  $o_\mathsf{g} = \left\langle \begin{matrix} \mathsf{pre}(o) \cup \langle g_{grs}/0\rangle \cup \langle g_{net}/0\rangle, \\ \mathsf{eff}(o) \cup \langle g_{grs}/1\rangle \cup \langle g_{net}/1\rangle, c(o) \end{matrix} \right\rangle$
- $\{o_\mathsf{mid} \mid o \in O, u(o) > 0\}$ *where*
  $o_\mathsf{mid} = \langle \mathsf{pre}(o) \cup \langle srch/1\rangle, \mathsf{eff}(o), c(o)\rangle$
- $\{o^+ \mid o \in O, u(o) > 0, \mathsf{eff}(o) \cap \mathsf{improved}(s) = \emptyset\}$ *where*
  $o^+ = \left\langle \begin{matrix} \mathsf{pre}(o) \cup \langle end/1\rangle \cup \langle g_{net}/0\rangle, \\ \mathsf{eff}(o) \cup \langle end/0\rangle \cup \langle g_{net}/1\rangle, c(o) \end{matrix} \right\rangle$
- $\{o_\mathsf{nm}, o_\mathsf{m} \mid o \in O, u(o) \leq 0, \mathsf{eff}(o) \cap \mathsf{improved}(s) \neq \emptyset\}$ *where;* $o_\mathsf{nm} = \langle \mathsf{pre}(o) \cup \langle g_{grs}/0\rangle, \mathsf{eff}(o), c(o)\rangle$, *and*
  $o_\mathsf{m} = \langle \mathsf{pre}(o) \cup \langle g_{grs}/0\rangle, \mathsf{eff}(o) \cup \langle g_{grs}/1\rangle, c(o)\rangle$
- $\{o_\mathsf{src} \mid o \in O, u(o) \leq 0, \mathsf{eff}(o) \cap \mathsf{improved}(s) = \emptyset\}$ *where*
  $o_\mathsf{src} = \langle \mathsf{pre}(o) \cup \langle srch/1\rangle, \mathsf{eff}(o), c(o)\rangle$
- $finish_\mathsf{g} = \left\langle \begin{matrix} \langle srch/1\rangle \cup \langle g_{grs}/1\rangle, \\ \langle get/1\rangle \cup \langle srch/0\rangle, c(finish) = 0 \end{matrix} \right\rangle$
  $finishG_\mathsf{g} = \left\langle \begin{matrix} \langle srch/1\rangle \cup \langle g_{grs}/1\rangle \cup \langle g_{net}/1\rangle, \\ \langle get/1\rangle \cup \langle srch/0\rangle, c(finishG) = 0 \end{matrix} \right\rangle$
- $\{o^{grs}_{ver}, o^{net}_{ver} \mid \langle v/d\rangle \in \mathsf{improved}(s)\}$ *where*
  $o^{grs}_{ver} = \left\langle \begin{matrix} \langle v/d\rangle \cup \langle get/1\rangle \cup \langle g_{net}/0\rangle \cup \langle end/0\rangle \cup \langle y^v_d/0\rangle, \\ \langle end/1\rangle \cup \langle get/0\rangle \cup \langle y^v_d/1\rangle, c(o^{grs}_{ver}) = 0 \end{matrix} \right\rangle$
  $o^{net}_{ver} = \left\langle \begin{matrix} \langle v/d\rangle \cup \langle y^v_d/1\rangle \cup \langle g_{net}/1\rangle, \\ \langle g_{net}/0\rangle \cup \langle y^v_d/0\rangle \cup \langle g/1\rangle, c(o^{net}_{ver}) = 0 \end{matrix} \right\rangle$

In plain words, $\Pi_\uparrow$ extends the structure of Π with a zero cost auxiliary actions and a zero utility auxiliary propositions. This auxiliary control structure allows for transformation of an OSP task to a certain classical planning task, that reflects the original OSP task properties in a way that they will be considered by the classical planning landmarks generation procedure. A valuable state for an OSP task Π with an initial state s_0 can be acquired with a set of events. Striving to extract refined landmarks of good quality, it will be beneficial to provide the landmark extraction procedure with as much as possible information about the events and the constraints that must hold in any valuable plan for an OSP task, that is, as accurate as possible auxiliary control structure. The auxiliary control structure constructed as follows:

- the propositions $srch, get, end, g_{grs}, g_{net}$ and g together with the actions $o^{grs}_{ver}, o^{net}_{ver}$ and $finish$ enforce the achievements with respect to constraints from Theorem 4,
- the goal proposition g can be achieved directly by a net positive utility action, which is also gross positive action,
- the artificial goal proposition g can be achieved by a net positive utility action, which is not a gross positive action, only after the verification phase is terminated, that is, the actions $O^{grs}_{ver}, O^{net}_{ver}$ and $finish$ applied. The verification phase is as follows: 1. regular actions can be applied only before $finish$ action, 2. O^{grs}_{ver} actions can be applied only after $finish$ action, 3. last occurrence of net utility actions (with g_{net} verification) can be applied only after O^{grs}_{ver} action, 4. goal achieving actions can be applied only with $o^{net}_{ver} \in O^{net}_{ver}$ action.

With this auxiliary control structure, the first part of any plan for $\Pi_\uparrow$ determines a plan for Π, the second part "verifies" that the end-state of that plan achieves a subset of valuable propositions from $\mathsf{improved}(s_0)$, the third part terminates the search with a net positive utility action. Finally, the last part grantees that $\mathsf{improved}(s) \cap s_0[\![\pi]\!] \neq \emptyset$ by verifying that the valuable proposition that initiated the verification phase, have not been lost during verification phase.

Constructing $\Pi_\uparrow$ from Π is trivially polynomial time, it allows for effective discovery of **value driven landmark** for Π using the standard machinery for landmark discovery.

Theorem 5. *For any OSP task* Π, *any landmark* L *for* $\Pi_\uparrow$ *such that* $L \subseteq O$ *is a value-landmark* L *for* Π.

With Theorem 5[1] in hand, we can derive ε-landmarks for Π using classical planning landmark extraction methods.

Incorporating the Compilation into the Planner. Based on the compilation described above, we modified the inc-compile-and-BFBB (Domshlak and Mirkis 2015) planner in two ways. First, we used the landmarks extracted from our compilation rather than the original compilation (Domshlak and Mirkis 2015). Second, instead of maintaining the set of *good* facts, we modify our landmark extraction technique to refer only to the best solution found so far, n^*. Specifically, gross positive actions in the second iteration and onward refer to an improvement relative to n^*, rather than to the initial state s_0.

Selective Action Split

In a SAS$^+$ action, a variable could have an effect defined on it without having a precondition. This makes computing the

[1]Due to space limitations, relegated to a full technical report.

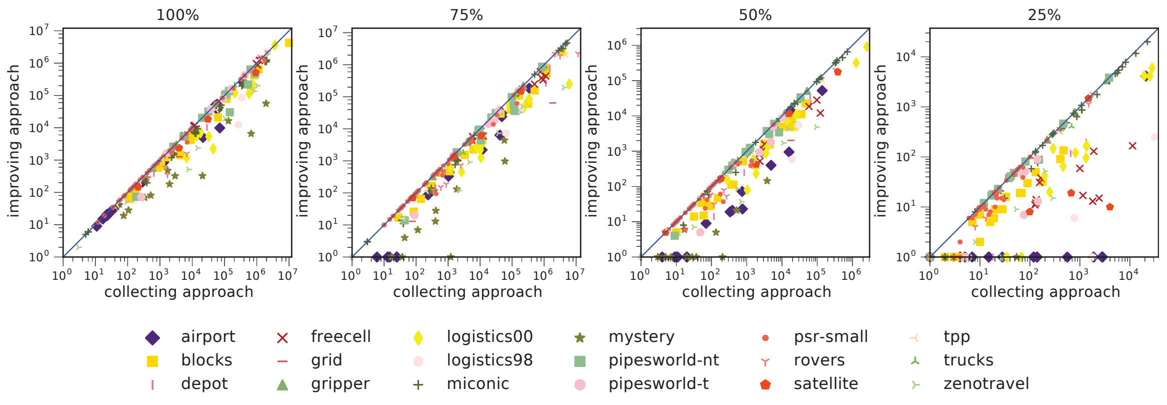


Figure 2: Comparative view of the number of expanded nodes for the fact oriented *collecting* approach (x-axis) vs. the action oriented *improving* approach (y-axis) on tasks with different budgets relative to c^*.

net utility of the action impossible, as the net utility depends on the state where the action is applied. In what follows we release our simplifying assumption of actions being represented in SAS.

Let us denote by S_a the set of states in which action a is applicable, and by S_a^+ the set of states in which a is applicable and has positive net utility. Since a is represented in $\textsc{sas}^+$, S_a is a Cartesian product of all possible propositions of the variables for which a has no preconditions, with a's preconditions. Let us also denote the set of variables which appear in a's effects but not in its preconditions by V_a. We could create multiple copies of a, one for each possible assignment to V_a, and compute for each of these whether it has positive net utility or not.

However, since we are only interested in checking whether the net utility is positive or not, we can potentially reduce the number of copies of a needed. Conceptually, we can think of a propositional formula expressed in DNF over the propositions associated with V_a, whose satisfying assignments are exactly those in which a has positive net utility, and another DNF formula whose satisfying assignments are those where a has non-positive net utility. Then it is enough to create a copy of a for each clause in these two DNF formulas, which guarantees that each copy always has either positive or non-positive net utility. Our implementation exploits mutual exclusion relations and cases where the utility of a variable is constant, to reduce the set of possible assignments even further, but we omit the exact details for the sake of brevity.

Empirical Evaluation

As OSP still lacks a standard suite of benchmarks for comparative evaluation, we have cast in this role the STRIPS classical planning tasks from the International Planning Competitions (IPC) 1998-2006. This “translation” to OSP was done by associating a separate value with each proposition in the corresponding classical IPC task.

In order to transform a classical planning task Π into a set of OSP tasks, we first solve Π optimally using Fast Downward (Helmert 2006) using the admissible landmarks heuristics (Karpas and Domshlak 2009) with a timeout of 10 minutes. Let us denote the cost of an optimal solution to Π by c^*. Similarly to (Domshlak and Mirkis 2015) we then generate several different OSP tasks, with budgets corresponding to 25%, 50%, 75%, and 100% of c^*. To complete the process, we must then assign a utility to each proposition in Π. We used 2 different methods of assigning utilities:

non-negative Each proposition $\langle v/d \rangle$ of multi-valued variable $v \in V$ was assigned a utility as follows:

$$u(\langle v/d \rangle) = \begin{cases} 2 & \text{if } \langle v/d \rangle \text{ is a goal proposition in } \Pi \\ 1 & \text{if } v \text{ is a goal variable in } \Pi \\ & \text{but } \langle v/d \rangle \text{ is not a goal proposition} \\ 0 & \text{otherwise} \end{cases}$$

negative Let s be the state reached by an optimal solution of Π. Then

$$u(\langle v/d \rangle) = \begin{cases} 1 & \text{if } \langle v/d \rangle \text{ is a goal proposition in } \Pi \\ -1 & \text{if } \langle v/d \rangle \in s \text{ but} \\ & \langle v/d \rangle \text{ is not a goal proposition} \\ 0 & \text{otherwise} \end{cases}$$

Non-negative Utility For the non-negative utility assignment method, we compared our approach, referred to as the *improving* approach here, to the *collecting* approach (Domshlak and Mirkis 2015). In both cases we ran inc-compile-and-BFBB with the blind heuristic, using our modified version of it for the improving approach.

We begin by comparing the informativeness of the landmarks of our *improving* approach, relative to that of the landmarks of the *collecting* approach. Figures 2a-2d compare the number of expanded nodes for each problem using collecting and improving approaches. As these figures show, the number of nodes expanded by the *improving* approach was always lower than the number of nodes expanded by the *collecting* approach. A closer look shows that as the budget decreases, our approach becomes more effective.

Table 1 shows the total number of problems solved by each approach for each budget. These results show the *improving* approach is able to solve more problems. Furthermore, in some cases, it is able to prove that there is no way

	25%				*50%*				*75%*				*100%*			
	Improving		*Collecting*		*Improving*		*Collecting*		*Improving*		*Collecting*		*Improving*		*Collecting*	
	Solved	*Time*	*Solved*	*Time*	*Solved*	*Time*	*Solved*	*Time*	*Solved*	*Time*	*Solved*	*Time*	*Solved*	*Time*	*Solved*	*Time*
airport (17)	17(**16**)	**44.85**	17(1)	97.31	17(**6**)	**230.08**	17(0)	256.95	17(**6**)	**248.97**	17(0)	352.57	17(0)	**365.72**	17(0)	443.78
blocks (18)	18(**3**)	**22.57**	18(0)	24.85	18(0)	**77.14**	18(0)	79.90	18(0)	**276.62**	18(0)	343.41	**18**(0)	**441.72**	16(0)	704.34
depot (4)	4(**2**)	**7.66**	4(0)	8.17	4(0)	**24.94**	4(0)	28.00	4(0)	**96.15**	4(0)	181.27	**4**(0)	**159.82**	3(0)	543.53
freecell (11)	11(0)	457.54	11(0)	**354.74**	11(0)	500.99	11(0)	**386.16**	11(0)	1500.16	11(0)	**1431.01**	11(0)	2143.73	11(0)	**1855.77**
grid (2)	2(2)	9.58	2(2)	**9.42**	2(1)	20.78	2(1)	**20.44**	2(0)	**39.67**	2(0)	121.06	2(0)	317.75	2(0)	**276.03**
gripper (4)	4(**1**)	8.35	4(0)	**5.96**	4(0)	31.16	4(0)	**21.79**	4(0)	32.55	4(0)	**21.75**	4(0)	62.37	4(0)	**46.68**
logistics00 (13)	13(**2**)	**15.49**	13(0)	20.33	**13**(0)	**103.77**	12(0)	189.02	**13**(0)	**751.91**	11(0)	1087.28	**13**(0)	611.25	11(0)	**594.93**
logistics98 (3)	3(**1**)	**2.97**	3(0)	5.68	3(0)	**6.58**	3(0)	8.22	3(0)	**518.47**	3(0)	534.32	3(0)	**544.24**	3(0)	587.24
miconic (47)	47(15)	69.34	47(15)	**58.60**	**47**(6)	383.90	46(6)	**361.32**	45(1)	1852.12	45(1)	**1710.02**	**44**(0)	**789.38**	40(0)	963.90
mystery (11)	11(**11**)	50.79	11(4)	**33.41**	11(**7**)	83.40	11(0)	**58.45**	11(**3**)	95.92	11(0)	**65.25**	11(0)	**128.36**	11(0)	264.23
pipesw-nt (13)	13(2)	101.74	13(2)	**89.53**	12(0)	155.44	12(0)	**144.90**	12(0)	**348.74**	12(0)	362.17	12(0)	**973.61**	12(0)	1180.86
pipesw-t (6)	6(**1**)	52.67	6(0)	**34.13**	6(0)	102.64	6(0)	**71.82**	6(0)	115.44	6(0)	**80.39**	6(0)	182.70	6(0)	**143.03**
psr-small (47)	47(**10**)	63.38	47(1)	**60.62**	47(0)	131.15	47(0)	**126.70**	47(0)	111.84	47(0)	**107.23**	47(0)	**364.56**	47(0)	384.18
rovers (7)	7(1)	10.52	7(1)	**10.20**	7(0)	**46.09**	7(0)	61.44	7(0)	**816.88**	7(0)	1295.53	**7**(0)	938.01	6(0)	**918.55**
satellite (6)	6(2)	**7.13**	6(2)	7.80	**6**(0)	**36.57**	5(0)	63.94	**5**(0)	23.64	4(0)	**22.60**	4(0)	**81.71**	4(0)	101.32
tpp (6)	6(5)	**3.26**	6(5)	3.56	7(1)	**20.30**	7(1)	23.12	6(1)	**151.68**	6(1)	310.71	6(0)	458.78	6(0)	**445.09**
trucks (2)	2(1)	4.76	2(1)	**3.88**	2(0)	14.26	2(0)	**12.78**	2(0)	23.04	2(0)	**20.09**	2(0)	41.12	2(0)	**34.52**
zenotravel (8)	8(**4**)	**8.31**	8(1)	10.79	8(1)	**28.68**	8(1)	32.29	8(1)	**36.13**	8(1)	194.32	8(0)	**206.90**	8(0)	575.28
total (225)	225(**79**)	940.91	225(35)	**838.98**	**224**(**22**)	1997.87	221(9)	**1947.24**	**221**(**12**)	**7039.93**	218(3)	8240.98	**219**(0)	**8811.73**	209(0)	10063.26

Table 1: Number of problems solved and (in brackets) solved with no search at all, across the different budgets using the *improving* approach and the *collecting* approach.

	25%			*50%*			*75%*			*100%*		
	Expanded	*Solved*	*Time*	*Expanded*	*Solved*	*Time*	*Expanded*	*Solved*	*Time*	*Expanded*	*Solved*	*Time*
airport 15)	0	15(15)	47.76	44599	15(6)	74.02	151860	15(6)	149.11	203372	15(0)	69.38
blocks 17)	405	17(4)	21.17	24379	17(1)	32.09	1945724	17(1)	144.81	14218611	17(1)	706.26
depot 3)	25	3(2)	11.42	16928	3(1)	17.38	382154	3(0)	43.46	13900	2(0)	8.88
freecell 5)	48	5(1)	77.51	2463	5(0)	122.81	14644	5(0)	132.95	24733	5(0)	141.51
grid 2)	0	2(2)	35.49	2021	2(1)	36.76	1142555	2(0)	111.88	1409996	2(0)	399.31
gripper 3)	821	3(1)	3.27	10258	3(0)	6.84	34214	3(0)	10.25	60207	3(0)	14.25
logistics00 10)	1043	10(2)	14.41	54470	10(0)	22.84	479866	10(0)	40.51	1854165	10(0)	108.12
logistics98 2)	9	2(1)	1.97	10957	2(0)	3.98	94961	2(0)	8.23	327929	2(0)	21.95
miconic 40)	12746	40(15)	44.23	639797	40(6)	134.43	4748596	40(1)	425.22	11337706	40(0)	933.18
mystery 11)	0	11(11)	49.48	160	11(10)	51.71	12516	11(5)	67.39	395511	11(0)	121.13
pipesw-nt 11)	250	11(2)	48.19	21490	11(0)	99.09	290079	11(0)	178.4	2248028	11(0)	677.66
pipesw-t 5)	71	5(1)	45.68	6386	5(0)	71.05	95533	5(0)	88.71	356736	5(0)	115.91
rovers 4)	15	4(1)	2.6	7563	4(0)	5.58	2637	4(0)	5.18	10181	4(0)	7.21
satellite 4)	30	4(2)	4.53	13949	4(0)	9.97	187176	4(0)	24.52	1013108	0(0)	104.54
tpp 5)	0	5(5)	2.61	1932	5(1)	3.54	14455	5)	3.35	28278	5(0)	4.49
trucks 2)	730	2(1)	2.92	21946	2(0)	6	77382	2(0)	9.62	82969	2(0)	10.88
zenotravel 7)	34	7(4)	11.94	10594	7(1)	19.94	229144	7(1)	30.14	1477266	7(1)	85.49
total 146)	16227	146(70)	425.18	882892	146(27)	718.03	9903496	146(15)	1473.72	35062696	145(2)	3630.15

Table 2: Performance of the utility setting independent OSP solver across the different budgets, on domains with negative utility setting, in terms of expanded nodes. The number of problems that were solved with no search is in brackets.

to improve upon the initial state with the given budget — this happens when the total cost of the value-landmarks discovered exceeds the budget. In this case, there is no need to perform search; the total number of problems for each domain where this occurs is in parentheses. Additionally, the table shows the total planning time for the set of commonly solved problems in each domain, where again the *improving* approach is faster on average.

Negative Utility For this utility assignment method, we could not compare to the *collecting* approach, since it is restricted to non-negative utilities. Thus, we only evaluated the *improving* approach using our modified version of inc-compile-and-BFBB.

Table 2 shows the results of the *improving* approach on these problems. While we have nothing to compare the results with, our planner does manage to optimally solve all but 1 of the tasks, and manages to solve quite a few of them with no search.

Conclusion and Future Work

We have addressed OSP with general additive utility functions. By defining the notions of net utility of an action, and of a gross positive action, we were able to exploit observations about the structure of optimal plans to derive value-landmarks. These landmarks can then be used in an OSP planner to obtain state-of-the-art performance.

These notions are not restricted to OSP planning. For example, the notion of the net utility of an action can be adapted to classical planning, where an action has net positive utility if it achieves some goal facts without deleting any other goal facts. We believe exploiting this observation in the context of classical planning can be beneficial.

References

Bäckström, C., and Klein, I. 1991. Planning in polynomial time: the sas-pubs class. *Computational Intelligence* 7(3):181–197.

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

Bonet, B., and Helmert, M. 2010. Strengthening landmark heuristics via hitting sets. In *ECAI*, 329–334.

Coles, A. J., and Coles, A. 2011. Lprpg-p: Relaxed plan heuristics for planning with preferences. In *ICAPS*.

Domshlak, C., and Mirkis, V. 2015. Deterministic oversubscription planning as heuristic search: Abstractions and reformulations. *Journal of Artificial Intelligence Research* 52:97–169.

Domshlak, C.; Katz, M.; and Lefler, S. 2012. Landmark-enhanced abstraction heuristics. *Artificial Intelligence* 189:48–68.

Haddawy, P., and Hanks, S. 1993. *Utility models for goal-directed decision-theoretic planners*. University of Washington, Department if Computer Science and Engineering.

Haddawy, P., and Hanks, S. 1998. Utility models for goal-directed, decision-theoretic planners. *Computational Intelligence* 14(3):392–429.

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

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

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *Journal of Artificial Intelligence Research* 22:215–278.

Karpas, E., and Domshlak, C. 2009. Cost-optimal planning with landmarks. In *IJCAI*, 1728–1733.

Katz, M., and Domshlak, C. 2010. Optimal admissible composition of abstraction heuristics. *Artificial Intelligence* 174(12-13):767–798.

Mirkis, V., and Domshlak, C. 2013. Abstractions for oversubscription planning. In *ICAPS*.

Mirkis, V., and Domshlak, C. 2014. Landmarks in oversubscription planning. In *Proceedings of the Twenty-first European Conference on Artificial Intelligence*, 633–638. IOS Press.

Pommerening, F., and Helmert, M. 2013. Incremental lmcut. In *ICAPS*.

Smith, D. E. 2004. Choosing objectives in over-subscription planning. In *ICAPS*, volume 4, 393.

Vidal, V., and Geffner, H. 2006. Branching and pruning: An optimal temporal pocl planner based on constraint programming. *Artificial Intelligence* 170(3):298–335.

Zhu, L., and Givan, R. 2004. Heuristic planning via roadmap deduction. *IPC-4 Booklet* 64–66.