Journal of Artificial Intelligence Research 52 (2015) 97-169 Submitted 05/14; published 01/15

Deterministic Oversubscription Planning as Heuristic Search: Abstractions and Reformulations

Carmel Domshlak DCARMEL@IE.TECHNION.AC.IL
Vitaly Mirkis MIRKIS80@GMAIL.COM
Faculty of Industrial Engineering & Management,
Technion - Israel Institute of Technology,
Haifa, Israel

Abstract

While in classical planning the objective is to achieve one of the equally attractive goal states at as low total action cost as possible, the objective in deterministic oversubscription planning (OSP) is to achieve an as valuable as possible subset of goals within a fixed allowance of the total action cost. Although numerous applications in various fields share the latter objective, no substantial algorithmic advances have been made in deterministic OSP. Tracing the key sources of progress in classical planning, we identify a severe lack of effective domain-independent approximations for OSP.

With our focus here on optimal planning, our goal is to bridge this gap. Two classes of approximation techniques have been found especially useful in the context of optimal classical planning: those based on state-space *abstractions* and those based on logical *landmarks* for goal reachability. The question we study here is whether some similar-in-spirit, yet possibly mathematically different, approximation techniques can be developed for OSP. In the context of abstractions, we define the notion of additive abstractions for OSP, study the complexity of deriving effective abstractions from a rich space of hypotheses, and reveal some substantial, empirically relevant islands of tractability. In the context of landmarks, we show how standard goal-reachability landmarks of certain classical planning tasks can be compiled into the OSP task of interest, resulting in an equivalent OSP task with a lower cost allowance, and thus with a smaller search space. Our empirical evaluation confirms the effectiveness of the proposed techniques, and opens a wide gate for further developments in oversubscription planning.

1. Introduction

The tools of automated action planning allow autonomous systems selecting a course of action "to get things done." Deterministic planning is probably the most basic, and thus the most fundamental, setting of automated action planning (Russell & Norvig, 2009). It can be viewed as the problem of finding trajectories of interest in large-scale yet concisely represented state-transition systems. Computational approaches to deterministic planning vary around the way those "trajectories of interest" are defined.

The basic structure of acting in situations with underconstrained or overconstrained resources is respectively captured by what these days is called "classical" deterministic planning (Fikes & Nilsson, 1971), and by what Smith (2004) termed "oversubscription" deterministic planning (OSP). In classical planning, the task is to find the most *cost-effective* trajectory possible to a *goal-satisfying* state. In oversubscription planning, the task is to find the most *goal-effective* (or *valuable*) state possible via a *cost-satisfying* trajectory. In optimal classical planning and in optimal OSP, the tasks are further constrained to finding only *most* cost-effective trajectories and *most* goal-effective states, respectively. Classical planning and OSP can be viewed as foundational variants of deterministic planning, with many other variants, such as net-benefit planning and cost-bounded planning, being defined in terms of mixing and relaxing the two.[1]

While OSP has been extensively advocated over the years, the theory and practice of classical planning have been studied and advanced much more intensively. The remarkable success and continuing progress of heuristic-search solvers for classical planning is one notable example. Primary enablers of this success are the advances in domain-independent approximations, or heuristics, of the cost needed to achieve a goal state from a given state. It is thus possible that having a similarly rich palette of effective heuristic functions for OSP would advance the state of the art in that problem.

Two classes of approximation techniques have been found especially useful in the context of optimal classical planning: those based on state-space *abstractions* (Edelkamp, 2001; Haslum, Botea, Helmert, Bonet, & Koenig, 2007; Helmert, Haslum, Hoffmann, & Nissim, 2014; Katz & Domshlak, 2010a) and those based on logical *landmarks* for goal reachability (Karpas & Domshlak, 2009; Helmert & Domshlak, 2009; Domshlak, Katz, & Lefler, 2012; Bonet & Helmert, 2010; Pommerening & Helmert, 2013). Considering OSP as heuristic search, a question is then whether some similar-in-spirit, yet possibly mathematically different, approximation techniques can be developed for heuristic-search OSP. This is precisely the question we study here.

- Starting with the most basic question of what state-space abstractions for OSP actually are, we show that the very notion of abstraction differs substantially for classical planning and OSP. Hence, first we define (additive) abstractions and abstraction heuristics for OSP. We then investigate the computational complexity of deriving effective abstraction heuristics in the scope of homomorphic abstraction skeletons, paired with cost, value, and budget partitions. Along with revealing some significant islands of tractability, this study exposes an interesting interplay between knapsack-style problems of combinatorial optimization, continuous convex optimization, and certain principles borrowed from explicit abstractions for classical planning.

- We introduce and study ε-landmarks, the logical properties of OSP plans that achieve valuable states. We show that ε-landmarks correspond to regular goal-reachability landmarks of certain classical planning tasks that can be straightforwardly derived from the OSP tasks of interest. We then show how such ε-landmarks can be compiled back into the OSP task of interest, resulting in an equivalent OSP task, but with a stricter cost satisfaction constraint, and thus with a smaller effective search space. Finally, we show how such landmark-based task enrichment can be combined in a mutually stratifying way with the best-first branch-and-bound search used for OSP planning, resulting in an incremental procedure that interleaves search and landmark discovery. The entire framework is independent of the OSP planner specifics, and in particular, of the heuristic functions it employs.

1. The connections and differences between some popular variants of deterministic planning are discussed in Section 2.

Our empirical evaluation on a large set of OSP tasks confirms the effectiveness of the proposed techniques. Moreover, to the best of our knowledge, our implementation constitutes the first domain-independent solver for optimal OSP, and we hope that more advances in this important computational problem will follow.

This work is a revision and extension of the formulations and results presented by the authors at ICAPS-2013 and ECAI-2014 (Mirkis & Domshlak, 2013, 2014). The paper is structured as follows. In Section 2 we formulate a general model of deterministic planning, define several variants of deterministic planning in terms of this model, and, in particular, show that oversubscription planning differs conceptually not only from classical planning, but also from other popular variants of deterministic planning such as net-benefit planning and cost-bounded planning. We also specify a simple model representation language for OSP, as well as provide the essential background on heuristic search, and, in particular, on OSP as heuristic search. Sections 3 and 4 are devoted, respectively, to abstractions and abstraction approximations for OSP. Section 5 is devoted to exploiting reachability landmarks in OSP tasks. In Section 6 we conclude and discuss some promising directions for future work. For the sake of readability, some of the proofs are relegated to Appendix A, and some details of the empirical results are relegated to Appendix B.

2. Background

As mentioned in the introduction, specific variants of deterministic planning differ in the way the interest and preference over trajectories are defined. For instance, in "classical planning" (Fikes & Nilsson, 1971), a trajectory is of interest if it connects a designated initial state to one of the designated goal states, with the preference being towards trajectories with lower total cost of the transitions along them. Among other, "non-classical" variants of deterministic planning are

- oversubscription planning (Smith, 2004), the topic of interest here;
- net-benefit planning (van den Briel, Sanchez, Do, & Kambhampati, 2004; Sanchez & Kambhampati, 2005; Baier, Bacchus, & McIlraith, 2009; Bonet & Geffner, 2008; Benton, Do, & Kambhampati, 2009; Coles & Coles, 2011; Keyder & Geffner, 2009);
- cost-bounded (also known as resource-constrained) planning (Haslum & Geffner, 2001; Hoffmann, Gomes, Selman, & Kautz, 2007; Gerevini, Saetti, & Serina, 2008; Thayer & Ruml, 2011; Thayer, Stern, Felner, & Ruml, 2012; Haslum, 2013; Nakhost, Hoffmann, & Müller, 2012); and
- planning with preferences over temporal properties of the trajectories (Baier et al., 2009; Gerevini, Haslum, Long, Saetti, & Dimopoulos, 2009; Benton, Coles, & Coles, 2012).

Interestingly, while working on this paper, we have learned that quite a few different variants of deterministic planning are often collectively referred to as "oversubscription planning". As a result, the difference between them in terms of expressiveness is not necessarily clear, and thus, the relationship between what we do here and what has already been done in the collective sense of "oversubscription planning" is not always apparent. This is the issue we will address first.

2.1 Models

Adopting and extending the notation of Geffner and Bonet (2013), we can view many variants of deterministic planning, including classical planning, as well as many popular non-classical variants, as special cases of a state model

$$M = \langle S, s_0, u, O, \varphi, c, Q \rangle \tag{1}$$

with:

- a finite set of states S,
- an initial state $s_0 \in S$,
- a state value function $u : S \mapsto \mathbb{R}^{0+} \cup \{-\infty\}$,
- operators $O(s) \subseteq O$ applicable in each state $s \in S$,
- a deterministic state transition function $\varphi(s, o)$ such that $s' = \varphi(s, o)$ stands for the state resulting from applying $o \in O(s)$ in s,
- an operator cost function $c : O \to \mathbb{R}^{0+}$, and
- a *quality* measure $Q : \mathcal{P} \mapsto \mathbb{R} \cup \{-\infty\}$, where $\mathcal{P}$ is the (infinite) set of trajectories from s_0 along operators O. A trajectory in $\mathcal{P}$ is a sequence of operators $\langle o_1, \dots, o_n \rangle$ such that $o_1 \in O(s_0)$ and, inductively, $o_i \in O(\varphi(\varphi(\dots \varphi(s_0, o_1) \dots, o_{i-2}), o_{i-1}))$.

In this model, *any* trajectory $\pi \in \mathcal{P}$ is a solution, with preference towards solutions of higher quality. In what follows, $s[\![\pi]\!]$ stands for the end-state of a trajectory π applied at state s, and $c(\pi) = \sum_{o \in \pi} c(o)$ is the additive cost of π. Likewise, the *graphical skeleton* $G_M = \langle S, T_\varphi, O \rangle$ of a model M refers to the edge-annotated, *unweighted* digraph induced by M where the nodes of G_M are the states S, the edge labels are the operators O, and T_φ contains an edge from s to s' labeled with o iff $o \in O(s)$ and $s' = \varphi(s, o)$.

First, consider a quality measure

$$Q^+(\pi) = u(s[\![\pi]\!]) - c(\pi). \tag{2}$$

This measure assumes that state values and operator costs are comparable, and thus represents a tradeoff between the value of the end-state and the cost of the trajectory. Consider now a fragment of the state model (1), instances of which all have the quality measure Q^+, and for each instance, the value function

$$u(s) = \begin{cases} \varepsilon, & s \in S_{goal} \\ -\infty, & \text{otherwise} \end{cases} \tag{3}$$

partitions the state space into $S_{goal} \subseteq S$, on which u takes a finite value $\varepsilon \geq 0$, and the rest of the states, on which u takes the value of $-\infty$. Finding an optimal solution for an instance M of this fragment corresponds to finding a *shortest path* from s_0 to a *single* node s_* in an edge-weighted digraph G, which is obtained from G_M by (i) annotating the edges of the latter with costs c, and (ii) adding a dummy node s_* and zero-cost edges from all

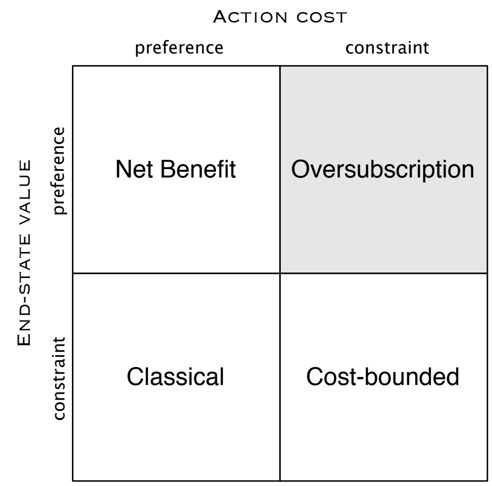


Figure 1: Schematic classification of four deterministic planning models along the strictness with which they approach the cost of operator sequences and the value of the operator sequence end-states. White blocks are for planning models that can be solved as single-source single-target shortest path problems.

goal nodes $s \in S_{goal}$ to s_*. While specified in a non-canonical way, this fragment can be easily verified to correspond to the model of *classical planning*, with S_{goal} being the classical planning goal states.

Staying with the quality measure Q^+ and removing now the requirement on u to comply with Eq. 3, we obtain a fragment that generalizes classical planning, and constitutes the basic model of what is called *net-benefit planning* (Sanchez & Kambhampati, 2005). Importantly, any instance M of this fragment can be reduced to finding a shortest path from a single node s_0 to a *single* node s_* in an edge-weighted digraph G, obtained from G_M by (i) annotating edges of G_M with costs c, (ii) adding a dummy node s_* and edges from all nodes $s \in S$ to s_*, and (ii) setting the cost of each such new edge (s, s_*) to $\sum_{s' \in S \setminus \{s\}} u(s')$. This reduction works because net-value maximization over the end state s is equivalent to minimization of the net-loss of giving up on all the other possible end states. This same basic idea underlies Keyder and Geffner's (2009) scheme for compiling certain standard representation formalisms for net-benefit planning into a standard classical planning formalism.[2]

Consider now an alternative quality measure

$$Q^b(\pi) = \begin{cases} u(s[\![\pi]\!]), & c(\pi) \leq b \\ -\infty, & \text{otherwise} \end{cases}, \tag{4}$$

2. It is worth noting that the wost-case complexity equivalence between classical planning and net-benefit planning has been shown prior to the work of Keyder and Geffner (2009) by van den Briel et al. (2004). However, this equivalence was not prescriptive enough to suggest practically effective compilations of compactly represented net-benefit planning tasks to classical planning tasks.

where $b \in \mathbb{R}^{0+}$ is a predefined bound on the cost of the trajectories. The fragment of the basic model, instances of which are characterized by having the quality measure Q^b and the "ε or $-\infty$" value functions as in Eq. 3, constitutes the model of what is called *cost-bounded planning* (Thayer & Ruml, 2011). Here as well, finding an optimal solution for a problem instance M corresponds to finding a shortest path from s_0 to s_* in an edge-weighted digraph G, which is derived from G_M identically to the case of classical planning.[3] This, in particular, explains why it is only natural for heuristic-search methods for cost-bounded planning to exploit heuristics developed for classical planning (Haslum, 2013).

We now arrive to a fourth fragment of the basic model. Staying with the quality measure Q^b and removing the requirement on u to comply with Eq. 3, we obtain a fragment that generalizes cost-bounded planning, and constitutes the model of *oversubscription planning* (Smith, 2004). As illustrated in Figure 1, the hard constraint of classical planning translates to soft preference in OSP, and the hard constraint of OSP translates to soft preference in classical planning. However, in contrast to cost-optimal, net-benefit, and classical planning, this fragment does not appear to be reducible to the single-source single-target shortest path problem. In terms of the digraph G obtained from G_M by annotating the edges with costs c, finding an optimal solution to an instance of oversubscription planning requires (i) finding shortest paths from s_0 to *all states* $s \in S$ with $u(s) > 0$, (ii) filtering out from these states those that are not reachable from s_0 within the cost allowance b, and (iii) selecting from the remaining states a state that maximizes u.

This contrast between oversubscription planning and the three other popular variants of deterministic planning discussed above has at least two important implications. First, while a single shortest path can be searched for using best-first forward search procedures such as A^*, searching for shortest paths to numerous targets simultaneously requires a different, more exhaustive, forward search framework such as branch-and-bound. Second, net-benefit and cost-bounded planning clearly have the potential to (directly or indirectly) reuse the rich toolbox of heuristic functions developed over the years for classical planning. In contrast, due to the differences in the underlying computational model, the same is not necessarily true for oversubscription planning, and examining the prospects of heuristic functions in OSP is precisely the focus of our work here.

2.2 Notation

For $k \in \mathbb{N}^+$, by $[k]$ we denote the set $\{1, 2, \ldots, k\}$. The indicator function of a subset A of a set X is a function $\mathbb{1}_A : X \to \{0, 1\}$ defined as $\mathbb{1}_A(x) = 1$ if $x \in A$ and $\mathbb{1}_A(x) = 0$ if $x \notin A$. Following Nebel (2000), when we talk about the *size* of a mathematically well-defined object x, symbolically $||x||$, we mean the size of a (reasonable) encoding of x. An assignment of a variable v to value d is denoted by $\langle v/d \rangle$; we often refer to such single variable assignments as *propositions*.

3. Strictly speaking, once a shortest path π from s_0 to s_* is found, it should still be checked against the cost bound b. This test, however, is local to π, and problem solving finishes independently of the test's outcome.

2.3 Model Representation

Departing from a very general model of oversubscription planning, in what follows we restrict our attention to instances of that model that are compactly representable in a language close to the SAS$^+$ language for classical planning (Bäckström & Klein, 1991; Bäckström & Nebel, 1995). In this language, a deterministic **oversubscription planning (OSP) task** is given by a sextuple

$$\Pi = \langle V, s_0, u; O, c, b \rangle, \tag{5}$$

where

(1) $V = \{v_1, \dots, v_n\}$ 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;

(2) $s_0 \in S$ is a designated *initial state*;

(3) u is an efficiently computable *state value* function $u : S \to \mathbb{R}^{0+}$;

(4) O is a finite set of *operators*, with each operator $o \in O$ 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;

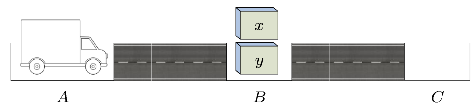
(5) $c : O \to \mathbb{R}^{0+}$ is an *operator cost* function;

(6) $b \in \mathbb{R}^{0+}$ is a *cost budget* allowed for the task.

Now consider the semantics of such a task description in terms of our basic model. An OSP task $\Pi = \langle V, s_0, u; O, c, b \rangle$ can be said to induce the model $M_\Pi = \langle S, s_0, u, O, \varphi, c, Q^b \rangle$, with Q^b being the quality measure (4) instantiated with the Π's budget b, and the transition function φ being specified as follows. 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. Similarly to the classical planning semantics of SAS$^+$, operator o is applicable in a state s iff $s[v] = \mathsf{pre}(o)[v]$ 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 $s[\![o]\!]$. This notation is only defined if o is applicable in s. Denoting an empty sequence of operators by ϵ, applying a sequence of operators $\langle o_1, \dots, o_m \rangle$ to a state s is defined inductively as $s[\![\epsilon]\!] := s$ and $s[\![o_1, \dots, o_j]\!] := s[\![o_1, \dots, o_{j-1}]\!][\![o_j]\!]$. An operator sequence π is called an *s-plan* if it is applicable in state s and $Q^b(\pi) \neq -\infty$, that is, $c(\pi) \leq b$.

Some auxiliary notation is used later on: For an OSP task $\Pi = \langle V, s_0, u; O, c, b \rangle$, by $\mathcal{D} = \bigcup_{v \in V} dom(v)$ we denote the union of the (uniquely labeled) state-variable domains. For a state s and a proposition $\langle v/d \rangle \in \mathcal{D}$, $\langle v/d \rangle \in s$ is used as a shortcut notation for $s[v] = d$.

An example of a simple OSP task in Figure 2 is used to illustrate this model representation. In this example, a truck is initially at location A, and it can drive (only) from location A to location B and from location B to location C. Two packages, x and y, are initially at location B. When a package and the truck are in the same location, the package can be loaded onto the truck, and when a package is on the truck, it can be unloaded at the

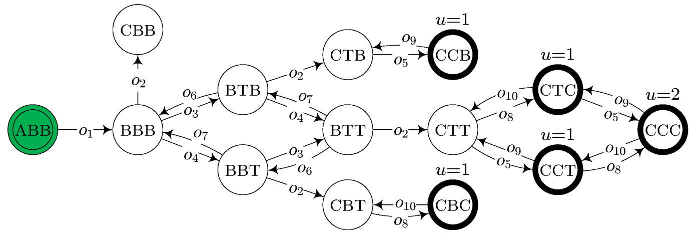


(a)

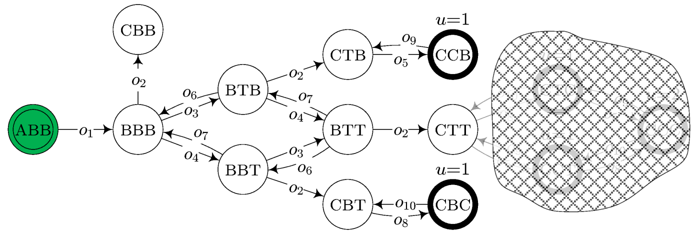
o_i	drive_{AB} $i = 1$	drive_{BC} $i = 2$	load_{Bx} $i = 3$	load_{By} $i = 4$	unload_{Cx} $i = 5$
$\mathsf{pre}(o_i)$	$\{\langle t/A \rangle\}$	$\{\langle t/B \rangle\}$	$\{\langle t/B \rangle, \langle x/B \rangle\}$	$\{\langle t/B \rangle, \langle y/B \rangle\}$	$\{\langle t/C \rangle, \langle x/T \rangle\}$
$\mathsf{eff}(o_i)$	$\{\langle t/B \rangle\}$	$\{\langle t/C \rangle\}$	$\{\langle x/T \rangle\}$	$\{\langle y/T \rangle\}$	$\{\langle x/C \rangle\}$

o_i	unload_{Bx} $i = 6$	unload_{By} $i = 7$	unload_{Cy} $i = 8$	load_{Cx} $i = 9$	load_{Cy} $i = 10$
$\mathsf{pre}(o_i)$	$\{\langle t/B \rangle, \langle x/T \rangle\}$	$\{\langle t/B \rangle, \langle y/T \rangle\}$	$\{\langle t/C \rangle, \langle y/T \rangle\}$	$\{\langle t/C \rangle, \langle x/C \rangle\}$	$\{\langle t/C \rangle, \langle y/C \rangle\}$
$\mathsf{eff}(o_i)$	$\{\langle x/B \rangle\}$	$\{\langle y/B \rangle\}$	$\{\langle y/C \rangle\}$	$\{\langle x/T \rangle\}$	$\{\langle y/T \rangle\}$

(b)



(c)



(d)

Figure 2: A simple running example of an OSP task, with (a) illustrating the story, (b) listing the operators, and (c)-(d) depicting the graphical skeleton of the induced state model; (c) shows the region of the graphical skeleton G_{M_Π} that is structurally reachable from the initial state ABB, and the grayed area in (d) corresponds to the sub-region that cannot be reached from the initial state under the budget $b = 4$.

truck's current location. Each (drive, load, and unload) operator in the task costs one unit of cost, and the cost budget is set to four units of cost. Finally, a value of one (value unit) is earned for each package present at location C.

This OSP task Π is described here using three state variables $V = \{t, x, y\}$, with $dom(t) = \{A, B, C\}$ and $dom(x) = dom(y) = \{A, B, C, T\}$, corresponding to the possible locations of the truck and the two packages, respectively. The operator set $O = \{o_1, \ldots, o_{10}\}$ is detailed in Figure 2(b). In the state model $M_\Pi = \langle S, s_0, u, O, \varphi, c, Q^b \rangle$ induced by this

```
BFBB (Π = ⟨V, s_0, u; O, c, b⟩)
  open := new max-heap ordered by f(n) = h(s⟨n⟩, b − g(n))
  initialize best solution n* := make-root-node(s_0)
  open.insert(n*)
  closed:= ∅;
  best-cost:= 0
  while not open.empty()
          n := open.pop-max()
          if f(n) ≤ u(s⟨n*⟩): break
          if u(s⟨n⟩) > u(s⟨n*⟩): update n* := n
          if s⟨n⟩ ∉ closed or g(n) < best-cost(s⟨n⟩):
             closed:= closed ∪ {s⟨n⟩}
             best-cost(s⟨n⟩) := g(n)
             foreach o ∈ O(s⟨n⟩):
                      n′ := make-node(s⟨n⟩⟦o⟧, n)
                      if g(n′) > b or f(n′) ≤ u(s⟨n*⟩): continue
                      open.insert(n′)
  return n*
```

Figure 3: Best-first branch-and-bound (*BFBB*) search for OSP

task, we have $S = dom(t) \times dom(x) \times dom(y)$, the initial state $s_0 = \text{ABB}$ (with the three letters in the names of the states capturing the three components of the domain cross-product), operator cost $c(o_i) = 1$ for all operators o_i, cost budget $b = 4$, and state values

$$u(s) = \begin{cases} 1, s \in \{\star\text{AC}, \star\text{BC}, \star\text{CA}, \star\text{CB}\} \\ 2, s \in \{\star\text{CC}\} \\ 0, \text{otherwise} \end{cases} .$$

The graphical skeleton G_{M_Π} is depicted in Figures 2(c) and 2(d): Figure 2(c) shows the region of the graphical skeleton G_{M_Π} that is structurally reachable from the initial state ABB, and the grayed area in Figure 2(d) corresponds to the sub-region that cannot be reached from the initial state under the budget $b = 4$.

2.4 OSP as Heuristic Search

The two major ingredients of any heuristic-search planner are its search algorithm and heuristic function. In classical planning, the heuristic is typically a function $h : S \to \mathbb{R}^{0+} \cup \{\infty\}$, with $h(s)$ estimating the cost $h^*(s)$ of optimal s-plans. A heuristic h is **admissible** if it is *lower-bounding*, that is, $h(s) \leq h^*(s)$ for all states s. All common heuristic search algorithms for optimal classical planning, such as A^*, require admissible heuristics.

In contrast, a heuristic in OSP is a function $h : S \times \mathbb{R}^{0+} \to \mathbb{R}^{0+}$, with $h(s, b)$ estimating the value $h^*(s, b)$ of optimal s-plans under cost budget b. A heuristic h is **admissible** if it is *upper-bounding*, that is, $h(s, b) \geq h^*(s, b)$ for all states s and all cost budgets b. Here as well, search algorithms for optimal OSP, such as best-first branch-and-bound (*BFBB*),[4] require admissible heuristics for pruning search branches without violating solution optimality.

Figure 3 depicts a pseudo-code description of *BFBB* for OSP; $s\langle n\rangle$ there denotes the state associated with search node n, and cost-so-far $g(n)$ is the total cost of the action sequence associated with n. Unlike in A^*, the order in which the nodes are selected from the OPEN list does not affect the optimality guarantees (though it may, of course, seriously affect the empirical efficiency of the search). In Figure 3, the ordering of OPEN corresponds to the decreasing order of $h(s\langle n\rangle, b - g(n))$. The duplicate detection and reopening mechanisms in *BFBB* are similar to those in A^* (Pearl, 1984). In addition, *BFBB* maintains the best solution n^* found so far and uses it to prune all generated nodes evaluated no higher than $u(s\langle n^*\rangle)$. Likewise, complying with the semantics of OSP, all generated nodes n with cost-so-far $g(n)$ higher than the problem's budget b are also immediately pruned. When the OPEN list becomes empty or the node n selected from the list promises less than the lower bound, *BFBB* returns (the plan associated with) the best solution n^*. If h is admissible, that is, the h-based pruning of the generated nodes is sound, then the returned plan is guaranteed to be optimal.

Let us now return to the heuristic functions. In domain-independent planning they should be automatically derived from the description of the model in the language of choice. A useful heuristic function must be both efficiently computable from the description of the model, as well as relatively accurate in its estimates. Improving the accuracy of a heuristic function without substantially worsening the time complexity of computing it translates into faster search for plans.

In classical planning, numerous approximation techniques, such as monotonic relaxation (Bonet & Geffner, 2001, 2001; Hoffmann & Nebel, 2001), critical trees (Haslum & Geffner, 2000), network flow (van den Briel, Benton, Kambhampati, & Vossen, 2007; Bonet, 2013), logical landmarks for goal reachability (Richter, Helmert, & Westphal, 2008; Karpas & Domshlak, 2009; Helmert & Domshlak, 2009; Bonet & Helmert, 2010), and abstractions (Edelkamp, 2001; Helmert, Haslum, & Hoffmann, 2007; Katz & Domshlak, 2010a), have been translated to effective heuristic functions. Likewise, different heuristics for classical planning can also be combined into their point-wise maximizing and/or additive ensembles (Edelkamp, 2001; Haslum, Bonet, & Geffner, 2005; Coles, Fox, Long, & Smith, 2008; Katz & Domshlak, 2010b; Helmert & Domshlak, 2009).

In contrast, development of heuristic functions for OSP has not progressed beyond the initial ideas of Smith (2004). In principle, the reduction of Keyder and Geffner (2009) from net-benefit to classical planning can be used to reduce OSP to classical planning with *real-valued* state variables (Koehler, 1998; Helmert, 2002; Fox & Long, 2003; Hoffmann, 2003; Gerevini, Saetti, & Serina, 2003; Gerevini et al., 2008; Edelkamp, 2003; Dvorak & Barták, 2010; Coles, Coles, Fox, & Long, 2013). So far, however, progress in heuristic-search classical planning with numeric state variables has mostly been achieved around direct extensions of delete relaxation heuristics via "numeric relaxed planning graphs" (Hoffmann, 2003; Edelkamp, 2003; Gerevini et al., 2003, 2008). Unfortunately, these heuristics do not preserve information on consumable resources such as budgeted operator cost in oversubscription

4. *BFBB* is also extensively used for net-benefit planning (Benton, van den Briel, & Kambhampati, 2007; Coles & Coles, 2011; Do, Benton, van den Briel, & Kambhampati, 2007), as well as some other variants of deterministic planning (Bonet & Geffner, 2008; Brafman & Chernyavsky, 2005).

planning: the "negative" action effects that decrease the values of numeric variables are ignored, possibly up to some special handling of so-called "cyclic resource transfer" (Coles et al., 2013).

As a first step in overcoming the lack of effective heuristics for OSP, in the next section we study *abstractions for OSP*, from their very definition and properties, to the prospects of deriving admissible abstraction heuristics. In Section 5 we then study the prospects of adapting to OSP the toolbox of logical landmarks for goal reachability. To date, abstractions and landmarks are responsible for most state-of-the-art admissible heuristics for classical planning, and thus are of special interest here.

3. Abstractions

The term "abstraction" is usually associated with simplifying the original model, factoring out details less crucial in the given context. Context determines which details can be reduced, which should better preserved, and how the abstraction is created and used (Cousot & Cousot, 1992; Clarke, Grumberg, & Peled, 1999; Helmert et al., 2014; Domshlak, Hoffmann, & Sabharwal, 2009; Katz & Domshlak, 2010b). In general terms, abstracting a model M corresponds to associating it with a set of (typically computationally more attractive) models $M_1, \ldots, M_k$ such that solutions to these models satisfy certain properties with respect to the solutions of M. In particular, in deterministic planning as heuristic search, abstractions are used to derive heuristic estimates for the states of the model of interest M: Given a state s of M and an abstraction $M_1, \ldots, M_k$,

(1) s is mapped to some "abstract states" $s_1 \in M_1, \ldots, s_k \in M_k$,

(2) the k models of the abstraction are solved for the respective initial states $s_1, \ldots, s_k$, and

(3) an aggregation of the quality of the resulting k solutions is used as the heuristic estimate for s.

Sometimes schematically and sometimes precisely, the process of constructing abstractions as above for a state model $M = \langle S, s_0, u, O, \varphi, c, Q \rangle$ can be seen as a two-step process of

(1) selecting an **abstraction skeleton** $\mathcal{AS} = \{(G_1, \alpha_1), \ldots, (G_k, \alpha_k)\}$, where each pair (G_i, α_i) comprises an edge-labeled digraph $G_i = \langle S_i, T_i, O_i \rangle$, with nodes S_i, edges T_i, and edge labels O_i, and a state mapping $\alpha_i : S \to S_i$, and

(2) extending $\mathcal{AS}$ to a set of **abstract models** $\mathcal{M} = \{M_1, \ldots, M_k\}$, such that, for $i \in [k]$, G_i is the graphical skeleton G_{M_i} of M_i.

To qualify as a valid abstraction of the model M, the resulting set of abstract models $\mathcal{M}$ should satisfy certain conditions specific to the variant of the deterministic planning under consideration. For instance, the optimal solutions of abstract models in classical planning are required to be at most as costly as the respective solutions in the original models, with that constraint to be satisfied by individual abstract models in case of max-aggregation (Pearl, 1984), or by the k abstract models jointly, in case of additive abstractions (Yang, Culberson, Holte, Zahavi, & Felner, 2008; Katz & Domshlak, 2010b). As we now show, the concept of abstractions in general, and additive abstractions in particular, is very different in OSP, and, for better or for worse, has many more degrees of freedom than the respective concepts in classical planning.

3.1 Abstractions of OSP Problems

Given an abstraction skeleton $\mathcal{AS} = \{(G_1, \alpha_1), \ldots, (G_k, \alpha_k)\}$ for an OSP state model $M = \langle S, s_0, u, O, \varphi, c, Q^b \rangle$, each digraph $G_i = \langle S_i, T_i, O_i \rangle$ implicitly defines a set of OSP state models consistent with it. This set is given by $C_i \times U_i \times B_i$, where C_i is the set of all functions from operators O_i to $\mathbb{R}^{0+}$, U_i is the set of all functions from states S_i to $\mathbb{R}^{0+}$, and $B_i = \mathbb{R}^{0+}$. In these terms, each point $(c, u, b) \in C_i \times U_i \times B_i$ induces an OSP model consistent with G_i, and vice versa.

Connecting between these sets of models for all the digraphs in $\mathcal{AS}$, let

$$\begin{aligned} \mathbf{C} &= C_1 \times \cdots \times C_k, \\ \mathbf{U} &= U_1 \times \cdots \times U_k, \\ \mathbf{B} &= B_1 \times \cdots \times B_k. \end{aligned}$$

For each state $s \in M$, every point $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in \mathbf{C} \times \mathbf{U} \times \mathbf{B}$ induces a set of models

$$\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})} = \left\{ M_1^{(\mathbf{c},\mathbf{u},\mathbf{b})}, \ldots, M_k^{(\mathbf{c},\mathbf{u},\mathbf{b})} \right\},$$

with $M_i^{(\mathbf{c},\mathbf{u},\mathbf{b})} = \left\langle S_i, \alpha_i(s_0), \mathbf{u}[i], O_i, \varphi_i, \mathbf{c}[i], Q^{\mathbf{b}[i]} \right\rangle$:

— the states S_i and operators O_i correspond to the nodes and edge labels of G_i;

— the transition function $\varphi_i(s, o) = s'$ iff T_i contains an arc from s to s' labeled with $o \in O_i$;

— the initial state $\alpha_i(s_0)$ is determined by the initial state s_0 and the state mapping α_i; and

— the operator cost function, state value function, and cost budget are all directly determined by the choice of $(\mathbf{c}, \mathbf{u}, \mathbf{b})$.

For some choices $(\mathbf{c}, \mathbf{u}, \mathbf{b})$ from $\mathbf{C} \times \mathbf{U} \times \mathbf{B}$, the induced sets of models $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}$ can be used for deriving admissible estimates for the state of interest s_0, while others cannot. The respective qualification is defined below.

Definition 1 (Additive OSP Abstraction)
Let $M = \langle S, s_0, u, O, \varphi, c, Q^b \rangle$ be an OSP model and $\mathcal{AS} = \{(G_1, \alpha_1), \ldots, (G_1, \alpha_k)\}$ be an abstraction skeleton for M. For $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in \mathbf{C} \times \mathbf{U} \times \mathbf{B}$, $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}$ is an **(additive) abstraction for** *M, denoted as*

$$\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})} \Subset_{\mathcal{AS}} M,$$

if and only if

$$h^*(s_0, b) \leq h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s_0, b) \stackrel{\mathsf{def}}{=} \sum_{i \in [k]} h_i^*(\alpha_i(s_0), \mathbf{b}[i]),$$

that is, when $h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s_0, b)$ is an admissible estimate of $h^*(s_0, b)$.

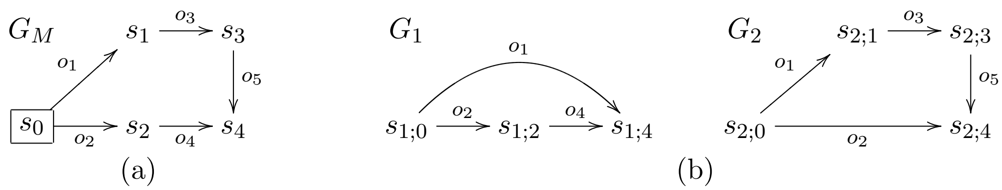


Figure 4: Illustration for our running example

In simple terms, a set of models forms an additive OSP abstraction if *jointly* the models in it do not *underestimate* the *value* that can be obtained from the *initial state*, within the given cost budget.[5] For example, let G_M in Figure 4a be the graphical skeleton of a state model $M = \langle \{s_0, \ldots, s_4\}, s_0, u, \{o_1, \ldots, o_5\}, \varphi, c, Q^b \rangle$, with $c(o_i) = 1$ for all operators o_i, $b = 2$, and $u(s_i) = \mathbb{1}_{\{4\}}(i)$. Let $\mathcal{AS} = \{(G_1, \alpha_1), (G_2, \alpha_2)\}$ be an abstraction skeleton for M, with G_1 and G_2 as in Figure 4b and with state mappings

$$\alpha_1(s_i) = \begin{cases} s_{1;4}, & i \in \{1,3\} \\ s_{1;i}, & \text{otherwise} \end{cases},$$

$$\alpha_2(s_i) = \begin{cases} s_{2;4}, & i = 2 \\ s_{2;i}, & \text{otherwise} \end{cases}.$$

Consider a set of models $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}$, with constant $\mathbf{c}[1](\cdot) = \mathbf{c}[2](\cdot) = 1$, $\mathbf{b}[1] = \mathbf{b}[2] = 2$, and, for $j \in [2]$, $\mathbf{u}[i](s_{i;j}) = \mathbb{1}_{\{4\}}(j)$. The optimal plan s_0-plan for M is $\pi = \langle (s_0, o_2, s_2), (s_2, o_4, s_4) \rangle$, with $Q^b(\pi) = 1$, while the optimal $\alpha_1(s_0)$-plan for $M_1^{(\mathbf{c},\mathbf{u},\mathbf{b})}$ is $\pi_1 = \langle (s_{1;0}, o_1, s_{1;4}) \rangle$, with $Q^{\mathbf{b}[1]}(\pi_1) = 1$, and the optimal $\alpha_2(s_0)$-plan for $M_2^{(\mathbf{c},\mathbf{u},\mathbf{b})}$ is $\pi_2 = \langle (s_{2;0}, o_2, s_{2;4}) \rangle$, with $Q^{\mathbf{b}[2]}(\pi_2) = 1$. Since

$$h^*(s_0, b) = Q^b(\pi) \leq Q^{\mathbf{b}[1]}(\pi_1) + Q^{\mathbf{b}[2]}(\pi_2) = h_1^*(\alpha_1(s_0), \mathbf{b}[1]) + h_2^*(\alpha_2(s_0), \mathbf{b}[2]),$$

$\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}$ is an additive abstraction for M.

Theorem 1 *For any OSP task $\Pi = \langle V, s_0, u; O, c, b \rangle$, any abstraction skeleton $\mathcal{AS} = \{(G_1, \alpha_1), \ldots, (G_k, \alpha_k)\}$ of M_Π, and any $\mathcal{M} \Subset_{\mathcal{AS}} M_\Pi$, if the digraphs of $\mathcal{AS}$ are given explicitly, then $h_{\mathcal{M}}(s_0, b)$ can be computed in time polynomial in $||\Pi||$ and $||\mathcal{M}||$.*

Proof: Let $\mathcal{M} = \{M_i\}_{i \in [k]}$, with $M_i = \langle S_i, \alpha_i(s_0), u_i, O_i, \varphi_i, c_i, Q^{b_i} \rangle$, be an additive abstraction for M_Π on the basis of $\mathcal{AS}$. For $i \in [k]$, let $S'_i = \{s \in S_i \mid c_i(\alpha_i(s_0), s) \leq b_i\}$. Since

5. In optimal classical planning, the requirement for the abstraction to not overestimate the costs is typically posed for all the states of the original model, not just for the initial state (Yang et al., 2008; Katz & Domshlak, 2010b; Helmert et al., 2014). This extra requirement, however, is for pragmatic reasons of efficiency as it allows the abstraction to be computed in preprocessing and not individually for every state examined by the search. Heuristics for OSP, however, are functions not only of the state but also of the available cost budget, and the latter directly applies to the initial (aka current) state only. In sum, while defining abstractions with respect to the entire state space is not a necessity in classical planning, in OSP it is not even clear whether defining abstractions with respect to a specific pair of a state and a budget can deliver any practical benefits. This should not, however, be interpreted as a formal impossibility claim, and further investigation in this direction is definitely worthwhile.

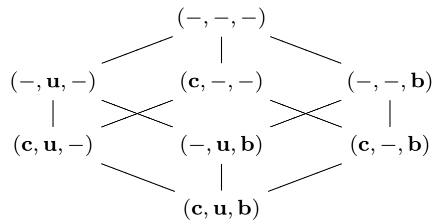


Figure 5: Fragments of restricted optimization over the abstractions $\mathbf{A} \subseteq \mathbf{C} \times \mathbf{U} \times \mathbf{B}$

the digraphs of $\mathcal{AS}$ are given explicitly, shortest paths from $\alpha_i(s_0)$ to all states in G_i (and thus, in particular, determining S'_i) can be computed in time polynomial in $||\mathcal{M}||$ for all $i \in [k]$. In turn, since $h^*_i(\alpha_i(s_0), b_i) = \max_{s \in S'_i} u_i(s)$, $h_{\mathcal{M}}(s_0, b) = \sum_{i \in [k]} h^*_i(\alpha_i(s_0), b_i)$ can be computed in time polynomial in $||\mathcal{M}||$. $\square$

The message of Theorem 1 is positive, yet it establishes only a necessary condition for the relevance of OSP abstractions in practice. Given an OSP task Π, and having fixed an abstraction skeleton $\mathcal{AS}$ with a joint performance measure space $\mathbf{C} \times \mathbf{U} \times \mathbf{B}$, we should be able to automatically separate between those $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in \mathbf{C} \times \mathbf{U} \times \mathbf{B}$ that constitute abstractions for M_Π and those that do not, and within the former set, denoted as

$$\mathbf{A} \subseteq \mathbf{C} \times \mathbf{U} \times \mathbf{B},$$

home in on an abstraction that provides us with as accurate (aka as *low*) an estimate of $h^*(s_0, b)$ as possible. Here, even the first item on the agenda is not necessarily trivial as, in general, $\mathbf{A}$ seems to lack convenient combinatorial properties. For instance, generally $\mathbf{A}$ does not form a combinatorial rectangle in $\mathbf{C} \times \mathbf{U} \times \mathbf{B}$: Consider the OSP state model G_M and abstraction skeleton $\mathcal{AS}$ from our running example. Let $\mathbf{c} \in \mathbf{C}$ be a cost function vector with both $\mathbf{c}[1]$ and $\mathbf{c}[2]$ being constant functions with value of 1, and two performance measures $(\mathbf{c}, \mathbf{u}, \mathbf{b}), (\mathbf{c}, \mathbf{u}', \mathbf{b}') \in \mathbf{C} \times \mathbf{U} \times \mathbf{B}$ defined via budget vectors $\mathbf{b} = \{\mathbf{b}[1] = 2, \mathbf{b}[2] = 0\}$ and $\mathbf{b}' = \{\mathbf{b}'[1] = 0, \mathbf{b}'[2] = 2\}$, and value function vectors $\mathbf{u}$ and $\mathbf{u}'$, with $\mathbf{u}[1], \mathbf{u}[2], \mathbf{u}'[1]$, and $\mathbf{u}'[2]$ evaluating to zero on all states except for $\mathbf{u}[1](s_{1;4}) = \mathbf{u}'[2](s_{2;4}) = 1$. It is not hard to verify that both $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})} \Subset_{\mathcal{AS}} M$ and $\mathcal{M}^{(\mathbf{c},\mathbf{u}',\mathbf{b}')} \Subset_{\mathcal{AS}} M$: In $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}$, the state $s_{1;4}$ with $\mathbf{u}[1](s_{1;4}) = 1$ is reachable in $M_1^{(\mathbf{c},\mathbf{u},\mathbf{b})}$ from $s_{1;0} = \alpha_1(s_0)$ under $\mathbf{b}[1] = 2$, while in $\mathcal{M}^{(\mathbf{c},\mathbf{u}',\mathbf{b}')}$, the state $s_{2;4}$ with $\mathbf{u}'[2](s_{2;4}) = 1$ is reachable in $M_2^{(\mathbf{c},\mathbf{u}',\mathbf{b}')}$ from $s_{2;0} = \alpha_2(s_0)$ under $\mathbf{b}'[2] = 2$. In contrast, both $\mathcal{M}^{(\mathbf{c},\mathbf{u}',\mathbf{b})} \not\Subset_{\mathcal{AS}} M$ and $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b}')} \not\Subset_{\mathcal{AS}} M$: In both sets of models, each model either comes with no budget (and the initial state in the model has the value of zero), or has no states with non-zero value at all. Hence, both $\mathcal{M}^{(\mathbf{c},\mathbf{u}',\mathbf{b})}$ and $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b}')}$ will estimate $h^*(s_0, b)$ as zero, while $h^*(s_0, b) = 1$.

In light of the above, we approach the overall agenda of complexity analysis of abstraction-based heuristic functions in steps, under different *fixations* of some of the three dimensions of $\mathbf{A}$: If, for instance, we are *given* a vector of value functions $\mathbf{u}$ that is *known* to belong to the projection of $\mathbf{A}$ on $\mathbf{U}$, then we can search for a quality abstraction from the abstraction subset $\mathbf{A}(-, \mathbf{u}, -) \subset \mathbf{A}$, corresponding to the projection of $\mathbf{A}$ on $\{\mathbf{u}\}$. As we show below, even some constrained optimizations of this kind can be challenging. The lattice in Figure 5 depicts the range of options for such constrained optimization; at the extreme settings,

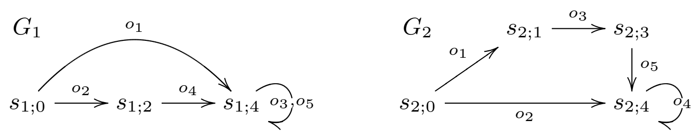


Figure 6: Homomorphic abstraction skeleton for $G(\Pi)$ in Figure 4

$\mathbf{A}(-,-,-)$ is simply a renaming of $\mathbf{A}$, and $\mathbf{A}(\mathbf{c},\mathbf{u},\mathbf{b})$ corresponds to a single abstraction $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})} \in \mathbf{A}$.

3.2 Partitions and Homomorphic Abstractions

We now proceed to consider a specific family of additive abstractions, reveal some of its interesting properties, and show that it contains substantial islands of tractability. With Definition 1 allowing for very general abstraction skeletons, in this work we focus on *homomorphic abstraction skeletons*[6] (Helmert et al., 2014).

Definition 2 *An abstraction skeleton $\mathcal{AS} = \{(G_1, \alpha_1), \ldots, (G_k, \alpha_k)\}$ for an OSP state model $M = \langle S, s_0, u, O, \varphi, c, Q^b \rangle$ is* **homomorphic** *if, for $i \in [k]$, $O_i = O$, and $\varphi(s, o) = s'$ only if $(\alpha_i(s), o, \alpha_i(s')) \in T_i$.*

For instance, in our running example, the abstraction skeleton in Figure 4b is not homomorphic (since, e.g., $(s_1, o_3, s_3) \in G_M$ yet $(\alpha_1(s_1), o_3, \alpha_1(s_3)) = (s_{1;4}, o_3, s_{1;4}) \notin G_{M_1}$), while the abstraction skeleton in Figure 6 is homomorphic. Furthermore, we focus on a fragment of additive abstractions

$$\mathbf{A}_p = \mathbf{A} \cap [\mathbf{C}_p \times \mathbf{U}_p \times \mathbf{B}_p],$$

where $\mathbf{C}_p \subseteq \mathbf{C}$, $\mathbf{U}_p \subseteq \mathbf{U}$, and $\mathbf{B}_p \subseteq \mathbf{B}$ correspond to *cost*, *value*, and *budget partitions*, respectively.

Definition 3 *Given an OSP state model $M = \langle S, s_0, u, O, \varphi, c, Q^b \rangle$, and a homomorphic abstraction skeleton $\mathcal{AS} = \{(G_1, \alpha_1), \ldots, (G_k, \alpha_k)\}$ for M with a joint performance measure $\mathbf{C} \times \mathbf{U} \times \mathbf{B}$,*

- $\mathbf{c} \in \mathbf{C}$ *is a* **cost partition** *iff, for each operator $o \in O$, $\sum_{i \in [k]} \mathbf{c}[i](o) \leq c(o)$;*
- $\mathbf{u} \in \mathbf{U}$ *is a* **value partition** *iff, for each state $s \in S$, $\sum_{i \in [k]} \mathbf{u}[i](\alpha_i(s)) \geq u(s)$; and*
- $\mathbf{b} \in \mathbf{B}$ *is a* **budget partition** *iff, $\sum_{i \in [k]} \mathbf{b}[i] \leq b$.*

In what follows, for any node x of the lattice in Figure 5, by $\mathbf{A}_p(x)$ we refer to $\mathbf{A}(x) \cap \mathbf{A}_p$; e.g., $\mathbf{A}_p(-,\mathbf{u},-) = \mathbf{A}(-,\mathbf{u},-) \cap \mathbf{A}_p$.

We begin our analysis of $\mathbf{A}_p$ by establishing an interesting "completeness" relationship between the sets $\mathbf{C}_p$ and $\mathbf{B}_p$, as well as an even stronger individual "completeness" of $\mathbf{C}_p$ and $\mathbf{B}_p$. Formulated in Theorem 2, these properties of $\mathbf{A}_p$ play a key role in our computational analysis later on.

6. All the results also hold verbatim for the more general "labeled paths preserving" abstraction skeletons studied by Katz and Domshlak (2010b) in the context of optimal classical planning. However, the presentation is somewhat more accessible when restricted to homomorphic abstraction skeletons.

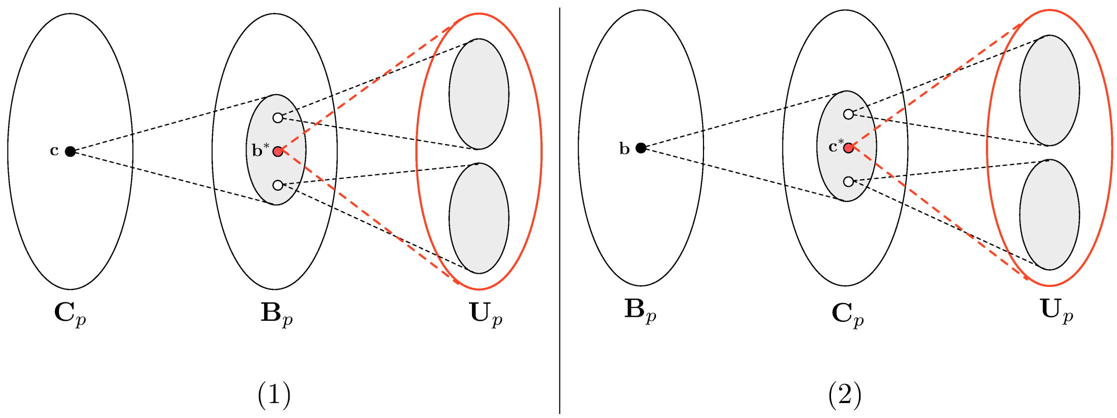


Figure 7: Illustration for sub-claims (1) and (2) of Theorem 2: In (1), the gray ellipse within $\mathbf{B}_p$ stands for the subset of budget partitions $\mathbf{b}$ that pair with $\mathbf{c}$ in some abstraction, that is, $\mathbf{A}_p(\mathbf{c}, -, \mathbf{b}) \neq \emptyset$. However, while pairing some of these budget partitions $\mathbf{b}$ with $\mathbf{c}$ requires then a careful selection of a value partition $\mathbf{u}$ (so that $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}$ will be an abstraction), there exists *some* budget partition $\mathbf{b}^*$ for which any choice of $\mathbf{u}$ will do the job.

Theorem 2 *Given an OSP task $\Pi = \langle V, s_0, u; O, c, b\rangle$ and a homomorphic abstraction skeleton $\mathcal{AS} = \{(G_1, \alpha_1), \dots, (G_k, \alpha_k)\}$ of M_Π,*

(1) for each cost partition $\mathbf{c} \in \mathbf{C}_p$, there exists a budget partition $\mathbf{b}^* \in \mathbf{B}_p$ such that $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b}^*)} \Subset_s \mathcal{AS}$ for all value partitions $\mathbf{u} \in \mathbf{U}_p$;

(2) for each budget partition $\mathbf{b} \in \mathbf{B}_p$, there exists a cost partition $\mathbf{c}^* \in \mathbf{C}_p$ such that $\mathcal{M}^{(\mathbf{c}^*,\mathbf{u},\mathbf{b})} \Subset_s \mathcal{AS}$ for all value partitions $\mathbf{u} \in \mathbf{U}_p$.

The proof of Theorem 2 appears in Appendix A, p. 145. Figure 7 illustrates the statement of sub-claim (1) of Theorem 2, as well as, indirectly, some of its corollaries.[7] The first corollary of Theorem 2 is that the projections of $\mathbf{A}_p$ on $\mathbf{C}_p$, $\mathbf{U}_p$, and $\mathbf{B}_p$ are the entire sets $\mathbf{C}_p$, $\mathbf{U}_p$, and $\mathbf{B}_p$, respectively. That is, any cost partition $\mathbf{c}$ (and similarly, any budget partition and any value partition) can be matched with an abstraction that has that partition as its component. Second, while not any budget partition $\mathbf{b}$ can be paired with a given cost partition $\mathbf{c}$ in abstractions for M_Π, that is, not for all $\mathbf{b} \in \mathbf{B}_p$, $\mathbf{A}_p(\mathbf{c}, -, \mathbf{b}) \neq \emptyset$, there are always *some* budget partitions that can be paired with $\mathbf{c}$. Finally, while pairing some of these "$\mathbf{c}$-compatible" budget partitions $\mathbf{b}$ with $\mathbf{c}$ requires then a careful selection of a value partition $\mathbf{u}$, there exists *some* "$\mathbf{c}$-compatible" budget partition $\mathbf{b}^*$ for which any choice of $\mathbf{u}$ will result in $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}$ being an abstraction of M_Π.

A priori, these properties of $\mathbf{A}_p$ should simplify the task of abstraction discovery and optimization within the space of partitions $\mathbf{C}_p \times \mathbf{U}_p \times \mathbf{B}_p$, and later we show that this is indeed the case. However, complexity analysis of abstraction discovery within $\mathbf{C}_p \times \mathbf{U}_p \times \mathbf{B}_p$ in most general terms is still problematic because the OSP formalism is parametric in

7. The respective illustration of sub-claim (2) of Theorem 2 is completely similar, *mutatis mutandis*.

the representation of value functions. Hence, here we proceed with examining abstraction discovery for OSP in the context of *fixed* value partitions $\mathbf{u} \in \mathbf{U}_p$.

4. From Value Partitions to Complete Abstractions

Let Π be an OSP task, $\mathcal{AS}$ be an explicitly given homomorphic abstraction skeleton of M_Π, and $\mathbf{u} \in \mathbf{U}_p$ be a value partition over $\mathcal{AS}$. An immediate corollary of Theorem 2 is that $\mathbf{A}_p(-, \mathbf{u}, -)$ is not empty, and thus we can try computing $\min_{(\mathbf{c},\mathbf{u},\mathbf{b}) \in \mathbf{A}_p(-,\mathbf{u},-)} h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s_0)$. As of yet, however, we do not know whether this task is polynomial-time solvable for any non-trivial class of value partitions. In fact, although $\mathbf{A}_p(-, \mathbf{u}, -)$ is known, by Theorem 2, to be non-empty, and so, too, are all of its subsets $\mathbf{A}_p(-, \mathbf{u}, \mathbf{b})$ and $\mathbf{A}_p(\mathbf{c}, \mathbf{u}, -)$, finding even just *any* abstraction $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in \mathbf{A}_p(-, \mathbf{u}, -)$ is not necessarily easy.

4.1 0-Binary Value Partitions

As a first step, we now examine abstraction discovery within a fragment of $\mathbf{A}_p$ in which all value functions $\mathbf{u}[i]$ of the abstract models are what we call *0-binary*. Later, in Section 4.2, we show how our findings for 0-binary abstract value functions can be extended to general value partitions.

Definition 4 *A real-valued function f is called* **0-binary** *if the codomain of f is $\{0, \sigma\}$ for some $\sigma \in \mathbb{R}^+$. A set F of 0-binary functions is called* **strong** *if all the functions in F have the same codomain $\{0, \sigma\}$.*

On the one hand, 0-binary functions constitute a rather basic family of value functions. Hence, if abstraction optimization is hard for them, it is likely to be hard for any non-trivial family of abstract value functions. On the other hand, 0-binary abstract value functions seem to fit well abstractions of planning tasks in which value functions are linear combinations of indicators, each representing achievement of a “goal value” for some state variable.

In that respect, our first tractability results are for abstraction discovery in $\mathbf{A}_p(-, \mathbf{u}, -)$ where $\mathbf{u}$ is a *strong* 0-binary value partition. The first (and the simpler) result in Theorem 3 further assumes a fixed action cost partition, while the next result, in Theorem 7, is on simultaneous selection of admissible pairs of cost and budget partitions. In Corollary 4 and Theorem 10 we then show how the results of Theorem 3 and Theorem 7, respectively, can be extended to pseudo-polynomial algorithms for *general* 0-binary value partitions.

4.1.1 Strong 0-Binary Value Partitions and the Knapsack problem

Our first tractability result is for abstraction discovery within $\mathbf{A}_p(\mathbf{c}, \mathbf{u}, -)$ where $\mathbf{u}$ is a *strong* 0-binary value partition and $\mathbf{c}$ is an arbitrary cost partition. The key role here is played by the well-known Knapsack problem (Dantzig, 1930; Kellerer, Pferschy, & Pisinger, 2004). An instance $\left\langle \{w_i, \sigma_i\}_{i \in [n]}, W \right\rangle$ of the **Knapsack** problem is given by a weight allowance W and a set of objects $[n]$, with each object $i \in [n]$ being annotated with a weight w_i and a value σ_i. The objective is to find a subset $Z \subseteq [n]$ that maximizes $\sum_{i \in Z} \sigma_i$ over all subsets $Z' \subseteq [n]$ with $\sum_{i \in Z'} w_i \leq W$. By **strict Knapsack** we refer to a variant of Knapsack in which that inequality constraint is strict. Knapsack is NP-hard (Karp, 1972; Garey & Johnson, 1978), but there exist pseudo-polynomial algorithms for it that run in time polynomial in the description of the problem and in the unary representation of W (Dudzinski & Walukiewicz, 1987). The latter property makes solving Knapsack practical in many applications where the ratio $\frac{W}{\min_i w_i}$ is reasonably low. Likewise, if $\sigma_i = \sigma_j$ for all $i, j \in [n]$, then a greedy algorithm solves the problem in linear time by iteratively expanding Z by one of the weight-wise lightest objects in $[n] \setminus Z$, until Z cannot be expanded any further within W.

Theorem 3 ($\mathbf{A}_p(\mathbf{c}, \mathbf{u}, -)$ & strong 0-binary u)
Let $\Pi = \langle V, s_0, u; O, c, b\rangle$ be an OSP task, $\mathcal{AS}$ be an explicit homomorphic abstraction skeleton of M_Π, and $\mathbf{u} \in \mathbf{U}_p$ be a strong 0-binary value partition. Given a cost partition $\mathbf{c} \in \mathbf{C}_p$, finding an abstraction $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in \mathbf{A}_p(\mathbf{c}, \mathbf{u}, -)$ and computing the corresponding heuristic estimate $h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s_0, b)$ can be done in time polynomial in $||\Pi||$ and $||\mathcal{AS}||$.

Proof: The proof is by reduction to the polynomial fragment of the Knapsack problem corresponding to all items having identical value. Let $\mathcal{AS} = \{(G_1, \alpha_1), \dots, (G_k, \alpha_k)\}$, and, given that $\mathbf{u}$ is a strong 0-binary value partition, let the codomain of all $\mathbf{u}[i]$ be $\{0, \sigma\}$ for some $\sigma \in \mathbb{R}^+$.

For $i \in [k]$, let w_i be the cost of the cheapest path in G_i from $\alpha_i(s_0)$ to (one of the) states $s \in S_i$ with $\mathbf{u}[i](s) = \sigma$. Since $\mathcal{AS}$ is an *explicit* abstraction skeleton, the set $\{w_i\}_{i\in[k]}$ can be computed in time polynomial in $||\mathcal{AS}||$ using one of the standard algorithms for the single-source shortest paths problem. Consider now a Knapsack problem $\left\langle \{w_i, \sigma\}_{i\in[k]}, b\right\rangle$, with weights w_i being as above and value σ being identical for all objects. Let $Z \subseteq [k]$ be a solution to that (optimization) Knapsack problem; recall that it is computable in polynomial time. Given that, we define budget profile $\mathbf{b}^* \in \mathbf{B}$ as follows:

$$\text{for } i \in [k], \mathbf{b}^*[i] = \begin{cases} w_i, & i \in Z \\ 0, & \text{otherwise.} \end{cases}$$

What remains to be shown is that $(\mathbf{c}, \mathbf{u}, \mathbf{b}^*)$ actually induces an additive abstraction for M_Π. Assume to the contrary that $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b}^*)} \notin_{\mathcal{AS}} M_\Pi$, and let π be an optimal s_0-plan for Π. By the construction of our Knapsack problem and of $\mathbf{b}^*$, for each $i \in Z$, there is a $\alpha_i(s)$-plan π_i for $M_i^{(\mathbf{c},\mathbf{u},\mathbf{b}^*)}$ with $Q^{\mathbf{b}^*[i]}(\pi_i) = \sigma$. By Definition 1, our assumption implies that $Q^b(\pi) > \sum_{i\in Z} Q^{\mathbf{b}^*[i]}(\pi_i) = \sigma \cdot |Z|$. However, by Theorem 2, there exists at least one budget partition $\mathbf{b} \in \mathbf{B}_p$ such that $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})} \in_{\mathcal{AS}} M_\Pi$. Note that this budget partition induces a feasible solution $Z' = \{i \mid w_i \leq \mathbf{b}[i]\}$ for our Knapsack problem, satisfying $Q^b(\pi) \leq \sum_{i\in Z'} Q^{\mathbf{b}[i]}(\pi_i) = \sigma \cdot |Z'|$. This, however, implies $|Z| < |Z'|$, contradicting the optimality of Z, and thus accomplishing the proof that $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b}^*)} \in_{\mathcal{AS}} M_\Pi$. □

The construction in the proof of Theorem 3 may appear somewhat counterintuitive: while we are interested in minimizing the heuristic estimate of $h^*(s_0, b)$, the abstraction $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b}^*)}$ is selected via the value-maximizing Knapsack problem. Indeed, while ultimately we would like to obtain

$$\min_{\mathbf{b}\,:\,(\mathbf{c},\mathbf{u},\mathbf{b})\in\mathbf{A}_p} h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s_0, b), \tag{6}$$

the heuristic we manage to compute in polynomial time is actually

$$\max_{\mathbf{b}\,:\,(\mathbf{c},\mathbf{u},\mathbf{b})\in\mathbf{A}_p} h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s_0, b). \tag{7}$$

At the same time, note that, for a fixed pair of $\mathbf{c} \in \mathbf{C}_p$ and $\mathbf{u} \in \mathbf{U}_p$, this estimate in Eq. 7 is still at least as (and possibly much more) accurate as the estimate that would be obtained by providing each of the k abstract models with the entire budget b. Later we show that this superior accuracy is verified in our experiments, but first we proceed with examining working with general 0-binary value partitions.

While strong 0-binary value partitions are rather restrictive, finding an element of $\mathbf{A}_p(\mathbf{c}, \mathbf{u}, -)$ for general 0-binary $\mathbf{u}$ is no longer polynomial—a reduction from Knapsack is straightforward. However, Knapsack is solvable in pseudo-polynomial time, and plugging that Knapsack algorithm into the proof of Theorem 3 results in a search algorithm for $\mathbf{A}_p(\mathbf{c}, \mathbf{u}, -)$ with general 0-binary $\mathbf{u}$.

Corollary 4 ($\mathbf{A}_p(\mathbf{c}, \mathbf{u}, -)$ & 0-binary $\mathbf{u}$)
Let $\Pi = \langle V, s_0, u; O, c, b\rangle$ be an OSP task, $\mathcal{AS}$ be an explicit homomorphic abstraction skeleton of M_Π, and $\mathbf{u} \in \mathbf{U}_p$ be a 0-binary value partition. Given a cost partition $\mathbf{c} \in \mathbf{C}_p$, finding an abstraction $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in \mathbf{A}_p(\mathbf{c}, \mathbf{u}, -)$ and computing the corresponding heuristic estimate $h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s_0, b)$ can be done in time polynomial in $||\Pi||$, $||\mathcal{AS}||$, and the unary representation of the budget b of Π.

To test and illustrate the value that additive abstractions can bring to heuristic-search OSP, we implemented a prototype heuristic-search OSP solver[8] on top of the Fast Downward planner (Helmert, 2006). Since, unlike classical and net-benefit planning, 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 unit-value with each proposition in the conjunctive goal of the corresponding classical IPC task.

Within our prototype, we implemented the *BFBB* search for OSP, and provided support for some basic pattern-database abstraction skeletons, action cost partitions, and abstraction selection in $\mathbf{A}_p(\mathbf{c}, \mathbf{u}, -)$ for strong 0-binary value partitions as in the proof of Theorem 3. Specifically, for a task with k sub-goals:

(i) The abstraction skeleton comprised a set of k projections of the planning task onto connected subsets of ancestors of the respective k goal variables in the causal graph. The size of each projection was limited to 1000 abstract states, and the ancestors of the goal variable v were added to the corresponding projection (initialized to contain v) in a breadth-first manner, from v back along the arcs of the causal graph, until the abstraction could not be expanded within the aforementioned size limit.

(ii) The value partition $\mathbf{u}$ associated the entire value of each sub-goal $\langle v/d\rangle$ (only) with the projection associated with v.

(iii) The cost partition $\mathbf{c}$ distributed the cost of each operator o *uniformly* between all the projections that did not invalidate o, i.e., that reflected at least one state variable affected by o.

In our evaluation, we compared *BFBB* node expansions with three heuristic functions, tagged *blind*, *basic*, and $h_{\mathcal{M}}$. With all three heuristics, the h-value of a node n is set to 0 if the cost budget at n is over-consumed. If the cost budget is not over-consumed, then:

8. We are not aware of any other domain-independent planner for optimal OSP.

	25%			50%			75%			100%		
	$h_{\mathcal{M}}$	*basic*	*blind*	$h_{\mathcal{M}}$	*basic*	*blind*	$h_{\mathcal{M}}$	*basic*	*blind*	$h_{\mathcal{M}}$	*basic*	*blind*
airport (25)	23	23	23	20	20	20	19	20	18	19	20	18
blocks (23)	23	23	23	23	23	23	22	18	17	17	17	17
depot (3)	3	3	3	3	3	3	3	3	3	3	2	2
driverlog (12)	12	12	12	12	12	11	11	9	9	10	7	6
freecell (5)	5	5	5	5	5	5	5	5	5	5	5	5
grid (2)	2	2	2	2	2	2	2	2	2	1	1	1
gripper (6)	6	6	6	6	6	6	6	6	6	6	6	6
logistics (10)	10	10	10	10	10	10	10	10	10	10	10	10
miconic (50)	50	50	50	50	50	50	50	50	50	50	45	45
mystery (4)	4	4	4	4	4	4	4	4	4	3	3	2
openstacks (7)	7	7	7	7	7	7	7	7	7	7	7	7
rovers (10)	10	10	10	10	7	7	7	6	6	6	5	5
satellite (9)	9	8	8	7	6	6	6	4	5	5	4	4
tpp (7)	7	7	7	7	7	7	6	6	6	6	5	5
trucks (9)	9	9	9	9	8	8	6	5	5	5	5	5
pipesw-t (12)	12	12	12	12	12	12	12	11	11	11	10	10
pipesw-nt (7)	7	7	7	7	7	7	7	7	7	7	6	6
psr-small (30)	30	30	30	30	30	30	30	30	30	30	30	30
zenotravel (10)	10	10	10	10	9	8	9	8	8	8	7	7
total	239	238	238	234	228	226	222	211	209	209	195	191

Table 1: Number of problems solved across the different budgets using the OPEN list ordered by the heuristic evaluation as in Figure 3

- Blind *BFBB* constitutes a trivial baseline in which $h(n)$ is simply set to the total value of all goals.
- In *basic BFBB*, $h(n)$ is set to the total value of goals, each of which can be *individually* achieved within the respective projection abstraction (see Theorem 1) given the *entire remaining budget*.
- $h_{\mathcal{M}}$ is an additive abstraction heuristic that is selected from $\mathbf{A}_p(\mathbf{c}, \mathbf{u}, -)$ as in the proof of Theorem 3.

The evaluation contained all the planning tasks for which we could determine offline the minimal cost budget needed to achieve all the goals. Each such task was approached under four different budgets, corresponding to 25%, 50%, 75%, and 100% of the minimal cost needed to achieve all the goals in the task, and each run was restricted to 10 minutes. Table 1 shows the number of tasks solved within each domain for each level of cost budget.[9] Figure 8 depicts the results in terms of expanded nodes across the four levels of cost budget. (Figures 18-21 in Appendix B provide a more detailed view of the results in Figure 8 by breaking them down into different levels of cost budget.) Despite the simplicity of the abstraction skeletons we used, the number of nodes expanded by *BFBB* with $h_{\mathcal{M}}$ was typically substantially lower than the number of nodes expanded by *basic BFBB*, with the difference sometimes reaching three orders of magnitude.

9. We reiterate that a task is considered solved only upon the termination of *BFBB*, that is, when an optimal plan is found and proven to be optimal.

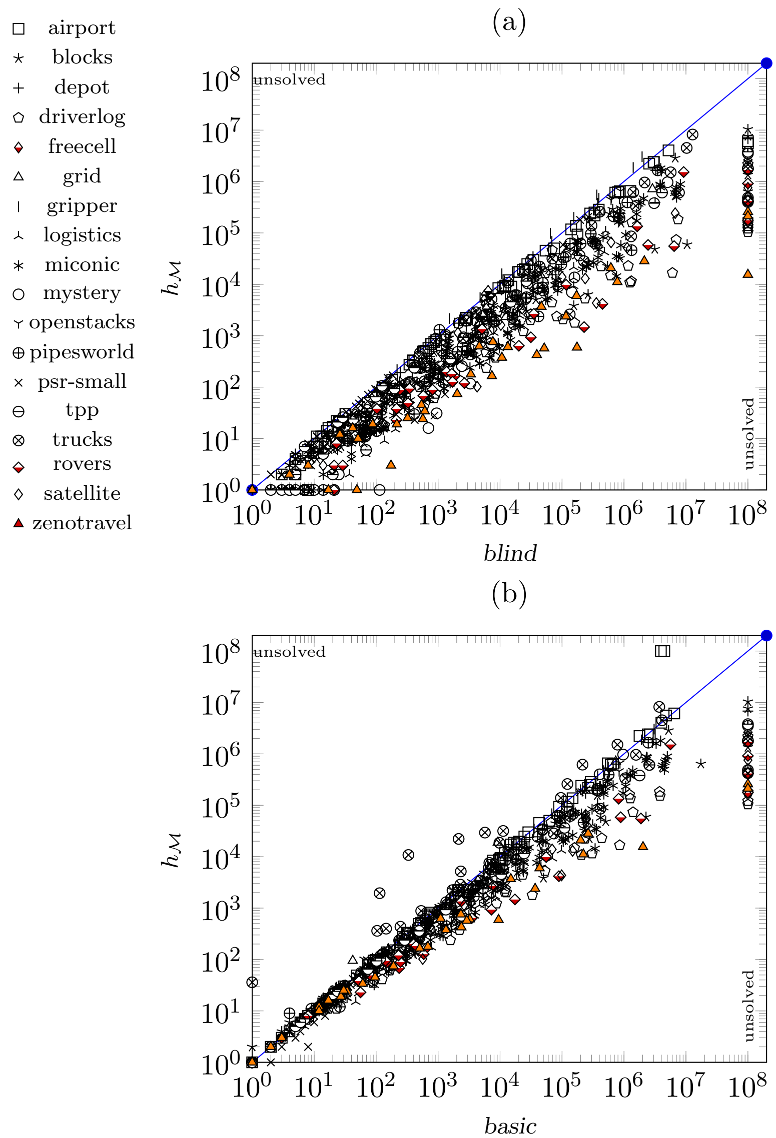


Figure 8: Comparative view of empirical results from Table 1 in terms of expanded nodes

4.1.2 Freeing Cost Partition: Knapsack Meets Convex Optimization

Returning now to the algorithmic analysis in the context of strong 0-binary value partitions, we now proceed with relaxing the constraint of sticking to a fixed action cost partition $\mathbf{c}$. This buys more flexibility in selecting abstractions from $\mathbf{A}_p(-, \mathbf{u}, -)$, allowing us to improve the accuracy of the heuristic estimates, while still retaining computational tractability.

```
input: Π = ⟨V, s_0, u; O, c, b⟩, AS = {(G_1, α_1), ..., (G_k, α_k)} of M_Π,
       strong 0-binary value partition u ∈ U_p
output: κ(u)

for i = 1 to k do
    reduce G_i to only nodes reachable from α_i(s_0)
for m = k downto 1 do
    if always-achievable(m) then return mσ
return 0

always-achievable(m):
  ellipsoid-method(separation-oracle-L_1^m) ↦ solution x ∈ dom(X) to L_1^m
  if x[ξ] ≤ b then return true
             else return false
```

(a)

```
separation-oracle-L_1^m(x ∈ dom(X)):
  let τ be a permutation of [k] such that
        x[𝕓[τ(1)]] ≤ x[𝕓[τ(2)]] ≤ ··· ≤ x[𝕓[τ(k)]]
  if x[ξ] ≤ Σ_{i∈[m]} x[𝕓[τ(i)]] then return Yes
                                  else return constraint ξ ≤ Σ_{i∈[m]} 𝕓[τ(i)]
```

(b)

Figure 9: A polynomial-time algorithm for computing $\kappa(\mathbf{u})$ for a strong 0-binary value partition $\mathbf{u} \in \mathbf{U}_p$ (Theorem 7)

Given an OSP task $\Pi = \langle V, s_0, u; O, c, b\rangle$, a homomorphic abstraction skeleton $\mathcal{AS}$, and a value partition $\mathbf{u} \in \mathbf{U}_p$ over $\mathcal{AS}$, let

$$\kappa(\mathbf{u}) = \min_{\mathbf{c}\in\mathbf{C}_p} \left[\max_{\mathbf{b}\,:\,(\mathbf{c},\mathbf{u},\mathbf{b})\in\mathbf{A}_p} h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s_0, b) \right]. \tag{8}$$

Obviously, the estimate $h(s_0, b) = \kappa(\mathbf{u})$ is at least as accurate as the estimate in Eq. 7 that is derived with respect to a fixed cost partition $\mathbf{c}$.

We now show that, for any OSP task Π, any abstraction skeleton $\mathcal{AS} = \{(G_1, \alpha_1), \ldots, (G_k, \alpha_k)\}$ of M_Π, and any strong 0-binary value partition $\mathbf{u} \in \mathbf{U}_p$ over $\mathcal{AS}$, $\kappa(\mathbf{u})$ can be computed in polynomial time. The corresponding algorithm is shown in Figure 9, with Figure 9a depicting the macro-flow of the algorithm and Figure 9b depicting the specific implementation of the solve sub-routine that makes the overall time complexity of the algorithm polynomial.

The high-level flow of the algorithm in Figure 9a is as follows. Since $\mathbf{u}$ is a strong 0-binary value partition, let the codomain of all the abstract value functions $\mathbf{u}[i]$ be $\{0, \sigma\}$ for some $\sigma \in \mathbb{R}^+$. Given that, for each $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in \mathbf{A}_p(-, \mathbf{u}, -)$, it holds that $h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s) = m\sigma$ for some $m \in \{0\} \cup [k]$. This is because each of the k abstract models in $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}$ can contribute to the additive estimate $h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s)$ either σ or 0.

The first loop of the algorithm is a preprocessing for-loop that eliminates from the abstraction skeleton all the nodes that are structurally unreachable from the abstract initial states $\alpha_1(s_0), \dots, \alpha_k(s_0)$.[10] For ease of presentation, in what follows we assume that this cleanup of the abstraction skeleton leaves each G_i with at least one state whose value is σ. The second for-loop of the algorithm decreasingly iterates over all the values $\{k\sigma, (k-1)\sigma, \dots, 2\sigma, \sigma\}$ that can possibly come from the abstractions in $\mathbf{A}_p(-, \mathbf{u}, -)$ as a positive estimate of $h^*(s_0, b)$. Each of these candidates for $\kappa(\mathbf{u})$ is tested in turn via the sub-routine always-achievable. If and when this test returns positive for the first time, then we are done, and the tested candidate $m\sigma$ is identified as $\kappa(\mathbf{u})$. Otherwise, if the test fails for all $m \in [k]$, then $\kappa(\mathbf{u}) = 0$, in particular implying that no state with value greater than 0 can be reached from s_0 under budget b.

The test of always-achievable for $\kappa(\mathbf{u}) = m\sigma$ is based on a linear program (LP) $\mathcal{L}_1^m$, given by Eq. 10. This linear program is defined over variables

$$\mathcal{X} = \{\xi\} \cup \bigcup_{i \in [k]} \left[\{d(s)\}_{s \in G_i} \cup \{\mathbb{b}[i]\} \cup \bigcup_{o \in O} \{\mathbb{c}[i](o)\} \right], \tag{9}$$

constraints (10a)-(10c), and the objective of maximizing the value of the variable ξ.

$$\begin{aligned}
&\mathcal{L}_1^m : \\
&\qquad \max \ \xi \\
&\text{subject to} \\
&\forall i \in [k] : \begin{cases} d(\alpha_i(s_0)) = 0, & \\ d(s) \le d(s') + \mathbb{c}[i](o), & \forall (s', o, s) \in G_i \\ \mathbb{b}[i] \le d(s), & \forall s \in G_i \text{ s.t. } \mathbf{u}[i](s) = \sigma \end{cases}, && (10a) \\
&\forall o \in O : \begin{cases} \mathbb{c}[i](o) \ge 0, & \forall i \in [k] \\ \sum_{i \in [k]} \mathbb{c}[i](o) \le c(o) & \end{cases}, && (10b) \\
&\forall Z \subseteq [k], |Z| = m : \ \xi \le \sum_{i \in Z} \mathbb{b}[i]. && (10c)
\end{aligned}$$

The roles of the different variables in $\mathcal{L}_1^m$ are as follows.

- Variable $\mathbb{c}[i](o)$ captures the cost to be associated with label o in the digraph G_i of $\mathcal{AS}$.
- For a state s in G_i, variable $d(s)$ captures the cost of the cheapest path in G_i from $\alpha_i(s_0)$ to s, *given* that the edges of G_i are weighted consistently with the values of the variables $\mathbb{c}[i](\cdot)$.
- Variable $\mathbb{b}[i]$ captures the minimal budget needed for reaching in G_i a state with value σ from state $\alpha_i(s_0)$, *given* that, again, the edges of G_i are weighted consistently with the variable vector $\mathbb{c}[i]$.

10. This preprocessing step can be replaced by adding some extra constraints in the linear program described below. However, that would unnecessarily complicate the presentation without adding much value.

- The singleton variable ξ captures the minimal total cost of reaching states with value σ in *precisely* m out of k models in $\mathcal{M}^{(\mathbb{c},\mathbf{u},\mathbb{b})}$.

The semantics of the constraints in $\mathcal{L}_1^m$ are as follows.

- The first two sets of constraints in (10a) come from a simple LP formulation of the single source shortest paths problem with the source node $\alpha_i(s_0)$: Optimizing $\sum_{i\in[k]}\sum_{s\in G_i} d(s)$ under a fixed weighting $\mathbb{c}$ of the edges leads to computing precisely that, for all k digraphs in $\mathcal{AS}$ simultaneously.

- The third set of constraints in (10a) establishes the costs of the cheapest paths in $\{G_i\}$ from states $\alpha_i(s_0)$ to states valued σ, enforcing the semantics of variables $\mathbb{b}[1],\dots,\mathbb{b}[k]$.

- Constraints (10b) are the cost partition constraints that enforce $\mathbb{c}\in\mathbf{C}_p$.

- Constraints (10c) enforce the aforementioned semantics of the objective variable ξ.

Two things are worth noting here. First, if all the nodes in the digraphs $G_1,\dots,G_k$ are structurally reachable from the "source nodes" $\alpha_1(s_0),\dots,\alpha_k(s_0)$, respectively (as it is ensured by the first for-loop of the algorithm), then the polytope induced by $\mathcal{L}_1^m$ is bounded and non-empty. Indeed, for any assignment to $\bigcup_{o\in O}\{\mathbb{c}[i](o)\}$ that is consistent with the positiveness constraints in (10b), all the variables $d(\cdot)$ are bounded from above by the lengths of the respective shortest paths. In turn, this bounding of $d(\cdot)$ bounds from above the variables $\mathbb{c}[1],\dots,\mathbb{c}[k]$ via the third set of constraints in (10a), and the constraints (10c) then bound from above the objective ξ.

Second, while the number of variables, as well as the number of constraints in (10a) and (10b), are polynomial in $||\Pi||$ and $||\mathcal{AS}||$, the number of constraints in (10c) is $\binom{k}{m}$. Thus, solving $\mathcal{L}_1^m$ using standard methods for linear programming is infeasible. In Lemma 5 below we show that this problem can actually be mitigated, and then, in Lemma 6 we show that the semantics of $\mathcal{L}_1^m$ match our objective of finding $\kappa(\mathbf{u})$.

Lemma 5 *The algorithm in Figure 9 terminates in time polynomial in* $||\Pi||$ *and* $||\mathcal{AS}||$.

Proof: The runtime complexity of the algorithm boils down to the complexity of solving $\mathcal{L}_1^m$, and, while the number of variables in $\mathcal{L}_1(m)$, as well as the number of constraints in (10a) and (10b), are polynomial in $||\Pi||$ and $||\mathcal{AS}||$, the number of constraints in (10c) is $\binom{k}{m}$. Thus, $\mathcal{L}_1^m$ cannot be solved in polynomial time using standard methods for linear programming, such as the Simplex algorithm (Dantzig, 1963) or the Interior-Point methods (Nemirovsky & Yudin, 1994). However, using some other algorithms, such as the Ellipsoid algorithm (Grotschel, Lovasz, & Schrijver, 1981) and the Random Walks family of algorithms originating in the work of Bertsimas and Vempala (2004), an LP with an exponential number of constraints can be solved in polynomial time, provided that we have a polynomial time "separation oracle" for that LP. A polynomial-time separating oracle for a convex set $K\subseteq\mathbb{R}^n$ is a procedure which given $x\in\mathbb{R}^n$, either verifies that $x\in K$ or returns a hyperplane separating x from K. The procedure should run in polynomial time. In our case, the separation problem is, given an assignment to the variables of $\mathcal{L}_1^m$, to test whether it satisfies (10a), (10b), and (10c), and if not, produce an inequality among (10a), (10b), and (10c) violated by that assignment.

We now show how our separation problem for $\mathcal{L}_1^m$ can be solved in polynomial time using what is called m-sum minimization LPs (Punnen, 1992), and this is precisely what the (parametrized with m) procedure separation-oracle-$\mathcal{L}_1^m$ in Figure 9b does. As the number of constraints in (10a) and (10b) is polynomial, their satisfaction by an assignment $\mathbf{x} \in dom(\mathcal{X})$ can be tested directly by substitution. For constraints (10c), let τ be a permutation of $[k]$ such that $\mathbf{x}[\mathbf{b}[\tau(1)]] \leq \mathbf{x}[\mathbf{b}[\tau(2)]] \leq \cdots \leq \mathbf{x}[\mathbf{b}[\tau(k)]]$. If $\mathbf{x}[\xi] \leq \sum_{i \in [m]} \mathbf{x}[\mathbf{b}[\tau(i)]]$, then it is easy to see that $\mathbf{x}$ satisfies *all* the constraints in (10c). Otherwise, we have our violated inequality $\xi \leq \sum_{i \in [m]} \mathbf{b}[\tau(i)]$. □

Lemma 6 *The algorithm in Figure 9a computes $\kappa(\mathbf{u})$.*

The proof of Lemma 6 appears in Appendix A, p. 146. Combining the statements of Lemmas 5 and 6, Theorem 7 summarizes our tractability result for abstraction discovery in $\mathbf{A}_p(-, \mathbf{u}, -)$ for strong 0-binary value partitions $\mathbf{u}$.

Theorem 7 ($\mathbf{A}_p(-, \mathbf{u}, -)(s)$ & strong 0-binary u)
Given an OSP task $\Pi = \langle V, s_0, u; O, c, b \rangle$, a homomorphic explicit abstraction skeleton $\mathcal{AS}$ of M_Π, and a strong 0-binary value partition $\mathbf{u} \in \mathbf{U}_p$, $\kappa(\mathbf{u})$ can be computed in time polynomial in $||\Pi||$ and $||\mathcal{AS}||$.

Unfortunately, the practical value of the result in Theorem 7 is yet to be evaluated. So far, we have not found a reasonably efficient implementation of the Ellipsoid method for linear inequalities, while, to the best of our knowledge, the Random Walks algorithms (Bertsimas & Vempala, 2004) have never been implemented at all. We hope that this state of affairs will change soon, allowing these powerful algorithms to be used not only in theory, but also in practice.

4.1.3 From Strong to General 0-Binary Value Partitions

Recall that the polynomial result of Theorem 3 for strong 0-binary value partitions easily extends in Corollary 4 to a pseudo-polynomial algorithm for general 0-binary value partitions. It turns out that a pseudo-polynomial extension of Theorem 7 is possible as well, though it is technically more involved. The corresponding algorithm is shown in Figure 10. Following the format of Figure 9, Figure 10a depicts the macro-flow of the algorithm and Figure 10b shows the specific implementation of the solve sub-routine by which the desired time complexity can be achieved.

Similarly to the algorithm in Figure 9, a preprocessing for-loop of the algorithm first eliminates from the abstraction skeleton all the nodes that are structurally unreachable from the abstract initial states $\alpha_1(s_0), \ldots, \alpha_k(s_0)$. Next, the algorithm performs a binary search over an interval containing $\kappa(\mathbf{u})$.[11] Since $\mathbf{u}$ is a 0-binary value partition, for $i \in [k]$, by $\{0, \sigma_i\}$, $\sigma_i \in \mathbb{R}^+$, we denote the codomain of the abstract value function $\mathbf{u}[i]$. Given that, for each $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in \mathbf{A}_p(-, \mathbf{u}, -)$, it holds that $h_{(\mathbf{c},\mathbf{u},\mathbf{b})}(s) = \sum_{i \in Z} \sigma_i$ for some $Z \subseteq [k]$. As the size of this combinatorial hypothesis space is prohibitive, the while-loop in Figure 10 performs a binary search over a relaxed hypothesis space, corresponding to the continuous

11. While a binary search could have been used in the algorithm in Figure 9 as well, there it would be a mere optimization, while here it is necessary to avoid an exponential blowup of the time complexity.

input: $\Pi = \langle V, s_0, u; O, c, b\rangle$, $\mathcal{AS} = \{(G_1, \alpha_1), \ldots, (G_k, \alpha_k)\}$ of M_Π,
    0-binary value partition $\mathbf{u} \in \mathbf{U}_p$
output: $\kappa_s(\mathbf{u})$

for $i = 1$ **to** k **do**
    reduce G_i to only nodes reachable from $\alpha_i(s_0)$
let $0 < \epsilon < \min_{i\in[k]} \sigma_i$
$\alpha \leftarrow 0$
$\beta \leftarrow \sum_{i\in[k]} \sigma_i$
while $\beta - \alpha > \epsilon$ **do**
    $v \leftarrow \alpha + (\beta - \alpha)/2$
    if always-achievable(v) **then** $\alpha \leftarrow v$
        else $\beta \leftarrow v$
if $\alpha = 0$ **then return** 0
    else return β

always-achievable(v):
  ellipsoid-method(separation-oracle-$\mathcal{L}_2^v$) $\mapsto$ solution $\mathbf{x} \in dom(\mathcal{X})$ to $\mathcal{L}_2^v$
  if $\mathbf{x}[\xi] \leq b$ **then return** true
    else return false

(a)

separation-oracle-$\mathcal{L}_2^v(\mathbf{x} \in dom(\mathcal{X}))$:
  strict-Knapsack($\langle \{\mathbf{x}[\mathbb{b}[i]], \sigma_i\}_{i\in[k]}, \mathbf{x}[\xi]\rangle$) $\mapsto$ solution $Z \subseteq [k]$
  if $\sum_{i\in Z} \sigma_i < v$ **then return** *Yes*
    else return constraint $\xi \leq \sum_{i\in Z} \mathbb{b}[i]$

(b)

Figure 10: A pseudo-polynomial algorithm for approximating $\kappa(\mathbf{u})$ for general 0-binary value partitions $\mathbf{u} \in \mathbf{U}_p$ (Theorem 10)

interval $[0, \sum_{i\in[k]} \sigma_i]$ of $\mathbb{R}^{+0}$. The parameter ϵ serves as the "sufficient precision" criterion for termination.

At an iteration corresponding to an interval $[\alpha, \beta]$, the algorithm uses its sub-routine always-achievable to test the hypothesis $\kappa(\mathbf{u}) \geq v$, where v is the mid-point of $[\alpha, \beta]$. If the test is positive, then the next tested hypothesis is $\kappa_s(\mathbf{u}) \geq v'$, where v' is the midpoint of $[v, \beta]$. Otherwise, the next hypothesis corresponds to the midpoint of $[\alpha, v)$. When the while-loop is done, the reported estimate is set to β; while there still might be some lag between β and $\kappa(\mathbf{u})$, this lag can be arbitrarily reduced by reducing ϵ, and anyway, $\beta \geq \kappa(\mathbf{u})$ ensures admissibility of the estimate. If, however, the while-loop terminates with $\alpha = 0$, then $\kappa(\mathbf{u}) \leq \beta \leq \epsilon < \min_{i\in[k]} \sigma_i$ implies $\kappa(\mathbf{u}) = 0$, and this is what we return.

The test of always-achievable for $\kappa(\mathbf{u}) \geq v$ is based on a linear program $\mathcal{L}_2^v$, which is defined over variables $\mathcal{X}$ as in Eq. 9, and is obtained from $\mathcal{L}_1^m$ by replacing constraints (10c) with constraints (11c):

$$\mathcal{L}_2^v:$$
$$\max \xi$$
subject to

$$\forall i \in [k]: \begin{cases} d(\alpha_i(s_0)) = 0, & \\ d(s) \leq d(s') + \mathbb{c}[i](o), & \forall (s', o, s) \in G_i \\ \mathbb{b}[i] \leq d(s), & \forall s \in G_i \text{ s.t. } \mathbf{u}[i](s) = \sigma_i \end{cases}, \tag{11a}$$

$$\forall o \in O: \begin{cases} \mathbb{c}[i](o) \geq 0, & \forall i \in [k] \\ \sum_{i\in[k]} \mathbb{c}[i](o) \leq c(o) & \end{cases}, \tag{11b}$$

$$\forall Z \subseteq [k] \text{ s.t.} \sum_{i\in Z} \sigma_i \geq v: \ \xi \leq \sum_{i\in Z} \mathbb{b}[i]. \tag{11c}$$

While the semantics of all variables but ξ remains as in $\mathcal{L}_1^m$, ξ now captures the minimal total cost of reaching some states $\{s_i\}_{i\in[k]}$ in the abstract models $\mathcal{M}^{(\mathbb{c},\mathbf{u},\mathbb{b})}$ such that the total value $\sum_{i\in[k]} \mathbf{u}[i](s_i) \geq v$. The new constraint (11c) enforces this semantics of ξ.

Lemma 8 *For any $\epsilon > 0$, the algorithm in Figure 10 terminates in time polynomial in $||\Pi||$, $||\mathcal{AS}||$, $\log \frac{1}{\epsilon}$, and a unary representation of the budget b of Π.*

Proof: The number of iterations of the while-loop is approximately $\log_2 \frac{\sum_{i\in[k]} \sigma_i}{\epsilon}$, and the run-time of each of its iterations boils down to the complexity of solving $\mathcal{L}_2^v$. Here, as in Lemma 5 with linear programs $\mathcal{L}_1^m$, the number of variables in $\mathcal{L}_2^v$, as well as the number of constraints in (11a) and (11b), are polynomial in $||\Pi||$ and $||\mathcal{AS}||$, while the number of constraints in (11c) is $\Theta(2^k)$. Therefore, $\textsf{always-achievable}(v)$ also employs the ellipsoid method with a sub-routine $\textsf{separation-oracle-}\mathcal{L}_2^v$ for the associated separation problem. We now show how that separation problem for $\mathcal{L}_2^v$ can be solved in pseudo-polynomial time using a standard pseudo-polynomial procedure for the *strict* Knapsack problem.

Given an assignment $\mathbf{x} \in dom(\mathcal{X})$, its feasibility with respect to (11a) and (11b) can be tested directly by substitution. For constraints (11c), let $Z \subseteq [k]$ be an optimal solution to the strict Knapsack problem $\langle \{\mathbf{x}[\mathbb{b}[i]], \sigma_i\}_{i\in[k]}, \mathbf{x}[\xi] \rangle$, with a weight allowance $\mathbf{x}[\xi]$ and k objects, with each object $i \in [k]$ being associated with weight $\mathbf{x}[\mathbb{b}[i]]$ and value σ_i.

- If the value $\sum_{i\in Z} \sigma_i$ of Z is *smaller* than v, then $\mathbf{x}$ satisfies *all* the constraints in (11c). Assume to the contrary that $\mathbf{x}$ violates some constraint in (11c), corresponding to a set $Z' \subseteq [k]$. By definition of (11c), $\sum_{i\in Z'} \sigma_i \geq v$, and by our assumption, $\mathbf{x}[\xi] > \sum_{i\in Z'} \mathbf{x}[\mathbb{b}[i]]$. That, however, implies that Z' is a feasible solution for our strict Knapsack, and of value higher than that of presumably optimal Z.

- Otherwise, if $\sum_{i\in Z} \sigma_i \geq v$, then Z itself provides us with a constraint in (11c) that is violated by $\mathbf{x}$. This is because $\mathbf{x}[\xi] > \sum_{i\in Z} \mathbf{x}[\mathbb{b}[i]]$ holds by the virtue of Z being a solution to the *strict* Knapsack problem $\langle \{\mathbf{x}[\mathbb{b}[i]], \sigma_i\}_{i\in[k]}, \mathbf{x}[\xi] \rangle$.

$\square$

Lemma 9 *For any $0 < \epsilon < \min_{i \in [k]} \sigma_i$, the algorithm in Figure 10a computes κ_ϵ such that $\kappa_\epsilon - \kappa(\mathbf{u}) \leq \epsilon$.*

The proof of Lemma 9 appears in Appendix A, p. 147. Combining the statements of Lemmas 8 and 9, Theorem 10 summarizes our result for optimized abstraction discovery in $\mathbf{A}_p(-, \mathbf{u}, -)$ for general 0-binary value partitions $\mathbf{u}$. Importantly, note that the algorithm in Figure 10 depends on the unary representation of only the budget, and not of the possible state values. In particular, it means that dependence of the complexity on the number of alternative sub-goals in the OSP task of interest is only polynomial. Finally, Theorem 10 is formulated in terms of the estimate precision only because the σ_i values of the abstract value functions $\mathbf{u}[i]$ can be arbitrary real numbers. In the case of integer-valued sets of functions $\mathbf{u}$, as well as in various special cases of real-valued functions, $\kappa(\mathbf{u})$ can be determined precisely using a simplification of the algorithm in Figure 10. For instance, if all $\sigma_1, \ldots, \sigma_k$ are integers, then setting ϵ to any value in $(0, 1)$ results in the while-loop terminating with $\alpha = \kappa(\mathbf{u})$. These details, however, are more of a theoretical interest; for reasonably small values of ϵ, in practice there will be no difference between estimates $h(s, b)$ and $h(s, b) + \epsilon$.

Theorem 10 ($\mathbf{A}_p(-, \mathbf{u}, -)(s)$ & 0-binary u)
Given an OSP task $\Pi = \langle V, s_0, u; O, c, b\rangle$, a homomorphic explicit abstraction skeleton $\mathcal{AS} = \{(G_1, \alpha_1), \ldots, (G_k, \alpha_k)\}$ of M_Π, a 0-binary value partition $\mathbf{u} \in \mathbf{U}_p$, and $\epsilon > 0$, it is possible to approximate $\kappa_s(\mathbf{u})$ within an additive factor of ϵ in time polynomial in $||\Pi||$, $||\mathcal{AS}||$, $\log \frac{1}{\epsilon}$, and a unary representation of the budget b of Π.

4.2 General Value Partitions

While 0-binary value partitions can be rather useful by themselves, it turns out that the pseudo-polynomial algorithms for abstraction discovery with explicit homomorphic abstraction skeletons and 0-binary value partitions can be extended rather easily to *arbitrary value partitions*, using the following observations:

(1) for any OSP task $\Pi = \langle V, s_0, u; O, c, b\rangle$, any homomorphic abstraction skeleton $\mathcal{AS} = \{(G_1, \alpha_1), \ldots, (G_k, \alpha_k)\}$ of M_Π, and any value partition $\mathbf{u}$ over $\mathcal{AS}$, the number of distinct values taken by $\mathbf{u}[i]$ is trivially upper-bounded by the number of states in G_i; and

(2) the pseudo-polynomial solvability of the Knapsack problem extends to its more general variant known as Multiple-Choice Knapsack (Dudzinski & Walukiewicz, 1987; Kellerer et al., 2004).

The **Multiple-Choice (MC) Knapsack** problem $\langle N_1, \ldots, N_m; W\rangle$ is given by a weight allowance W and m classes of objects $N_1, \ldots, N_m$, with each object $j \in N_i$ being annotated with a weight w_{ij} and a value σ_{ij}. The objective is to find a set Z that contains at most one object from each class and maximizes $\sum_{(i,j) \in Z} \sigma_{ij}$ over all such sets while satisfying $\sum_{(i,j) \in Z} w_{ij} \leq W$. By **strict MC-Knapsack**, we refer to a variant of MC-Knapsack in which that inequality constraint is strict. MC-Knapsack generalizes regular Knapsack and thus it is NP-hard. However, similarly to the regular Knapsack problem, MC-Knapsack also admits a pseudo-polynomial, dynamic programming algorithm that runs in time polynomial in the description of the problem and in the unary representation of W (Dudzinski & Walukiewicz, 1987; Kellerer et al., 2004).

Theorem 11 $(\mathbf{A}_p(\mathbf{c}, \mathbf{u}, -))$
Let $\Pi = \langle V, s_0, u; O, c, b\rangle$ *be an OSP task, let* $\mathcal{AS} = \{(G_1, \alpha_1), \ldots, (G_k, \alpha_k)\}$ *be an explicit homomorphic abstraction skeleton of* M_Π*, and let* $\mathbf{u} \in \mathbf{U}_p$ *be an arbitrary value partition over* $\mathcal{AS}$*. Given a cost partition* $\mathbf{c} \in \mathbf{C}_p$*, it is possible to find an abstraction* $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in \mathbf{A}_p(\mathbf{c}, \mathbf{u}, -)$ *and compute the corresponding heuristic estimate* $h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s_0, b)$ *in time polynomial in* $||\Pi||$*,* $||\mathcal{AS}||$*, and the unary representation of the budget* b*.*

Proof: The proof is very similar to the proof of Theorem 3, but with the compilation being to the MC-Knapsack problem.

For $i \in [k]$, let n_i be the number of distinct values taken by $\mathbf{u}[i]$, let $\{\sigma_{i1}, \ldots, \sigma_{in_i}\} \subset \mathbb{R}^+$ be the codomain of $\mathbf{u}[i]$, and, for $j \in [n_i]$, let w_{ij} be the cost of the cheapest path in G_i from $\alpha_i(s_0)$ to (one of the) states $s \in S_i$ with $\mathbf{u}[i](s) = \sigma_{ij}$. Since $\mathcal{AS}$ is an *explicit* abstraction skeleton, for $i \in [k]$, it holds that $n_i \leq |S_i|$, and the set $\{w_{ij}\}_{i\in[k],j\in[n_i]}$ can be computed in time polynomial in $||\mathcal{AS}||$ using one of the standard algorithms for the single-source shortest paths problem.

Consider now an MC-Knapsack problem with a weight allowance b and k classes of objects $N_1, \ldots, N_k$, with $|N_i| = n_i$ and each object $j \in N_i$ annotated with a weight w_{ij} and a value σ_{ij}. Let $Z \subseteq \bigcup_{i=1}^{k} N_i$ be a solution to that (optimization) MC-Knapsack problem; recall that it is computable in pseudo-polynomial time. Given that, we define budget profile $\mathbf{b}^* \in \mathbf{B}$ as follows:

$$\text{for } i \in [k], \mathbf{b}^*[i] = \begin{cases} w_{ij}, & (i,j) \in Z \\ 0, & \text{otherwise.} \end{cases}$$

Showing that $(\mathbf{c}, \mathbf{u}, \mathbf{b}^*)$ actually induces an additive abstraction for M_Π is completely identical to the proof of the corresponding argument in Theorem 3, and thus omitted. $\square$

Theorem 12 $(\mathbf{A}_p(-, \mathbf{u}, -))$
Given an OSP task $\Pi = \langle V, s_0, u; O, c, b\rangle$*, a homomorphic explicit abstraction skeleton* $\mathcal{AS} = \{(G_1, \alpha_1), \ldots, (G_k, \alpha_k)\}$ *of* M_Π*, an arbitrary value partition* $\mathbf{u} \in \mathbf{U}_p$ *over* $\mathcal{AS}$*, and* $\epsilon > 0$*, it is possible to approximate* $\kappa_s(\mathbf{u})$ *within an additive factor of* ϵ *in time polynomial in* $||\Pi||$*,* $||\mathcal{AS}||$*,* $\log \frac{1}{\epsilon}$*, and a unary representation of the budget* b *of* Π*.*

An algorithm for abstraction discovery as in Theorem 12 is depicted in Figure 11. Its high-level flow differs from the flow of the algorithm from Figure 10 for general 0-binary value partitions only in the initialization of parameters ϵ and β. The major difference between the algorithms is that here the tests of candidate values v are based on linear programs $\mathcal{L}_3^v$, which are defined as follows.

For $i \in [k]$, let $\{\sigma_{i1}, \ldots, \sigma_{in_i}\} \subset \mathbb{R}^+$ be the codomain of $\mathbf{u}[i]$. For $v \in \mathbb{R}^+$, the linear program $\mathcal{L}_3^v$ is defined in Eq. 13 over variables

$$\mathcal{X} = \{\xi\} \cup \bigcup_{i\in[k]} \left[\{d(s)\}_{s\in G_i} \cup \bigcup_{j\in[n_i]} \{\mathbb{b}[i,j]\} \cup \bigcup_{o\in O} \{\mathbb{c}[i](o)\} \right]. \tag{12}$$

input: $\Pi = \langle V, s_0, u; O, c, b\rangle$, $\mathcal{AS} = \{(G_1, \alpha_1), \ldots, (G_k, \alpha_k)\}$ of M_Π,
0-binary value partition $\mathbf{u} \in \mathbf{U}_p$
output: $\kappa_s(\mathbf{u})$

```
for i = 1 to k do
    reduce G_i to only nodes reachable from α_i(s_0)
let 0 < ε < min_{i∈[k]} min_{j∈[n_i]} σ_ij
α ← 0
β ← Σ_{i∈[k]} max_{j∈[n_i]} σ_ij
while β − α > ε do
    v ← α + (β − α)/2
    if always-achievable(v) then α ← v
                            else β ← v
if α = 0 then return 0
          else return β

always-achievable(v):
ellipsoid-method(separation-oracle-L_3^v) ↦ solution x ∈ dom(X) to L_3^v
  if x[ξ] ≤ b then return true
              else return false
```

(a)

```
separation-oracle-L_3^v(x ∈ dom(X)):
  strict-MC-Knapsack(⟨{x[b[1,j]], σ_1j}_{j∈[n_1]}, ..., {x[b[k,j]], σ_kj}_{j∈[n_k]}; x[ξ]⟩)
            ↦ solution Z ∈ [n_1] × ··· × [n_k]
  if Σ_{i∈[k]} σ_iZ(i) < v then return Yes
                          else return constraint ξ ≤ Σ_{i∈[k]} b[i, Z(i)]
```

(b)

Figure 11: (a) A modification of the algorithm from Figure 10 to arbitrary value partitions $\mathbf{u} \in \mathbf{U}_p$ (Theorem 12), and (b) a pseudo-polynomial time separation oracle for the corresponding linear programs $\mathcal{L}_3^v$ in Eq. 13

These variables differ from the variable set of $\mathcal{L}_2^v$ (see Eq. 9) by a larger set of $\mathbb{b}$-variables: Variable $\mathbb{b}[i,j]$ here captures the minimal budget needed for reaching in G_i a state with value $\sigma_{i,j}$ from state $\alpha_i(s_0)$, given that the edges of G_i are weighted consistently with the variable vector $\mathbb{c}[i]$.

$$\mathcal{L}_3^v:$$
$$\max\ \xi$$
subject to
$$\forall i \in [k]: \begin{cases} d(\alpha_i(s_0)) = 0, \\ d(s) \le d(s') + \mathbb{c}[i](o), & \forall (s', o, s) \in G_i \\ \mathbb{b}[i,j] \le d(s), & \forall j \in [n_i] \forall s \in G_i \text{ s.t. } \mathbf{u}[i](s) = \sigma_{ij} \end{cases}, \tag{13a}$$
$$\forall o \in O: \begin{cases} \mathbb{c}[i](o) \ge 0, & \forall i \in [k] \\ \sum_{i\in[k]} \mathbb{c}[i](o) \le c(o) \end{cases}, \tag{13b}$$
$$\forall Z \in [n_1] \times \cdots \times [n_k] \text{ s.t.} \sum_{i\in[k]} \sigma_{iZ(i)} \ge v: \ \xi \le \sum_{i\in[k]} \mathbb{b}[i, Z(i)]. \tag{13c}$$

Like what we had in Lemma 8 with linear programs $\mathcal{L}_2^v$, while the number of variables in $\mathcal{L}_3^v$, as well as the number of constraints in (13a) and (13b), are polynomial in $||\Pi||$ and $||\mathcal{AS}||$, the number of constraints in (13c) is $\Theta(d^k)$ where $d = \max_{i\in[k]} n_i$. Therefore, always-achievable(v) also employs the ellipsoid method with a pseudo-polynomial time separation oracle, but here the latter is based on solving a strict MC-Knapsack problem (see Figure 11b). Otherwise, solving $\mathcal{L}_2^v$ and solving $\mathcal{L}_3^v$ are similar.

Lemma 13 *For any $\epsilon > 0$, the algorithm in Figure 11 terminates in time polynomial in $||\Pi||$, $||\mathcal{AS}||$, $\log \frac{1}{\epsilon}$, and a unary representation of the budget b of Π.*

Lemma 14 *Given an OSP task $\Pi = \langle V, s_0, u; O, c, b\rangle$, a homomorphic explicit abstraction skeleton $\mathcal{AS} = \{(G_1, \alpha_1), \ldots, (G_k, \alpha_k)\}$ of M_Π, an arbitrary value partition $\mathbf{u} \in \mathbf{U}_p$ over $\mathcal{AS}$, and $\epsilon > 0$, the algorithm in Figure 11 computes κ_ϵ such that $\kappa_\epsilon - \kappa(\mathbf{u}) \le \epsilon$.*

The proof of Lemma 13 is similar to the proof of Lemma 8, with strict Knapsack separation problems being replaced with strict MC-Knapsack separation problems. The proof of Lemma 14 is also similar to the proof of Lemma 9, *mutatis mutandis.* Together, Lemmas 14 and 13 establish Theorem 12.

5. Landmarks in OSP

In addition to state-space abstractions, a family of approximation techniques that have been found extremely effective in the context of optimal classical planning is based on the notion of logical *landmarks for goal reachability* (Karpas & Domshlak, 2009; Helmert & Domshlak, 2009; Domshlak et al., 2012; Bonet & Helmert, 2010; Pommerening & Helmert, 2013). In this section we proceed with examining the prospects of such reachability landmarks in heuristic-search OSP planning.

5.1 Landmarks in Classical Planning

For a state s in a classical planning task Π, a landmark is a property of operator sequences that is satisfied by all s-plans (Hoffmann, Porteous, & Sebastia, 2004). For instance, a "fact landmark" for a state s is an assignment to a single variable that is true at some point in *every* s-plan. Most state-of-the-art admissible heuristics for classical planning use what are called **disjunctive action landmarks**, each corresponding to a set of operators such that every s-plan contains at least one operator from that set (Karpas & Domshlak, 2009; Helmert & Domshlak, 2009; Bonet & Helmert, 2010; Pommerening & Helmert, 2013). In what follows we consider this popular notion of landmarks, and simply refer to disjunctive action landmarks for a state s as *s-landmarks*. For ease of presentation, most of our discussion will take place in the context of landmarks for the initial state of the task, and these will simply be referred to as *landmarks (for Π)*.

Deciding whether an operator set $L \subset O$ is a landmark for classical planning task Π is PSPACE-hard (Porteous, Sebastia, & Hoffmann, 2001). Therefore, all landmark heuristics employ landmark discovery methods that are polynomial-time and sound, but incomplete. In what follows we assume access to such a procedure; the actual way the landmarks are discovered is tangential to our contribution. For a set $\mathcal{L}$ of s-landmarks, a **landmark cost function** $lcost : \mathcal{L} \to \mathbb{R}^{0+}$ is *admissible* if $\sum_{L \in \mathcal{L}} lcost(L) \leq h^*(s)$. For a singleton set $\mathcal{L} = \{L\}$, $lcost(L) := \min_{o \in L} c(o)$ is a natural admissible landmark cost function, and it extends directly to non-singleton sets of pairwise disjoint landmarks. For more general sets of landmarks, $lcost$ can be devised in polynomial time via operator cost partitioning (Katz & Domshlak, 2010b), either given $\mathcal{L}$ (Karpas & Domshlak, 2009), or within the actual process of generating $\mathcal{L}$ (Helmert & Domshlak, 2009).

5.2 ε-Landmarks and Budget Reduction

While landmarks play an important role in (both satisficing and optimal) classical planning, so far they have not been exploited in OSP. At first glance, this is probably no surprise, and not only because OSP has been investigated much less than classical planning: Since landmarks must be satisfied by all plans and because an empty operator sequence is always a plan for any OSP task, the notion of landmark does not seem useful here. Having said that, consider the anytime "output improvement" property of the *BFBB* forward search. The empty plan is not interesting there not only because it is useless, but also because it is "found" by *BFBB* right at the very beginning. In general, at all stages of the search, anytime search algorithms like *BFBB* maintain the best-so-far solution π, and prune all branches that promise value lower or equal to $Q^b(\pi)$. Hence, in principle, such algorithms may benefit from information about properties that are "satisfied by all plans with value larger than $Q^b(\pi)$." Polynomial-time discovery of such "value landmarks" for arbitrary OSP tasks is still an open problem. However, looking at what is needed and what is available, here we show that the classical planning machinery of reachability landmarks actually can be effectively exploited in OSP.

In what follows, we assume that the value function of Π is *additive*, with $u(s) = \sum_{\langle v/d \rangle \in s} u_v(d)$, with $u_v(d) \geq 0$ for all variable-value pairs $\langle v/d \rangle$. That is, the value of state s is the sum of the (mutually independent) non-negative marginal values of the propositions comprising s. With the value of different s-plans in an OSP task Π varying between zero and the value of the optimal s-plan (which may also be zero), let ε-**landmark** for state s be any property that is satisfied by any s-plan π that achieves *something valuable*. For instance, with the disjunctive action landmarks we use here, if $L \subseteq O$ is an ε-landmark for s, then every s-plan π with $Q^b(\pi) > 0$ contains an operator from L. In what follows, unless stated otherwise, we focus on ε-landmarks for (the initial state of) Π.

Definition 5 *Given an OSP task* $\Pi = \langle V, s_0, u; O, c, b\rangle$*, the* ε-**compilation** *of* Π *is a classical planning task* $\Pi_\varepsilon = \langle V_\varepsilon, s_{0\varepsilon}, G_\varepsilon; O_\varepsilon, c_\varepsilon\rangle$ *where*

$$
\begin{aligned}
V_\varepsilon &= V \cup \{g\},\\
&\quad \text{with } dom(g) = \{0,1\},\\
s_{0\varepsilon} &= s_0 \cup \{\langle g/0\rangle\},\\
G_\varepsilon &= \{\langle g/1\rangle\},\\
O_\varepsilon &= O \cup O_g = O \cup \left\{o_{\langle v/d\rangle} \mid \langle v/d\rangle \in \mathcal{D}, u_v(d) > 0\right\},\\
&\quad \text{with } \mathsf{pre}(o_{\langle v/d\rangle}) = \{\langle v/d\rangle\} \text{ and } \mathsf{eff}(o_{\langle v/d\rangle}) = \{\langle g/1\rangle\},\\
c_\varepsilon(o) &= \begin{cases} c(\omega), & \omega = o \in O\\ 0, & \omega = o_{\langle v/d\rangle} \in O_g \end{cases}.
\end{aligned}
$$

To put it simply, the semantics of the value $\langle g/1\rangle$ of the auxiliary variable g in Π_ε is that "it has been verified that some proposition with a positive value has been achieved". In these terms, Π_ε 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. Constructing Π_ε from Π is trivially polynomial time, and it allows us to discover ε-landmarks for Π using the standard machinery for classical planning landmark discovery.

Theorem 15 *For any OSP task* Π*, any landmark* L *for* Π_ε *such that* $L \subseteq O$ *is an* ε-*landmark for* Π.

Proof: The proof is rather straightforward. Let $\mathcal{P}$ be the set of all plans π for Π with $Q^b(\pi) > 0$ and $\mathcal{P}_\varepsilon$ the set of all plans for Π_ε. By the definition of $\mathcal{P}$, for any plan $\pi \in \mathcal{P}$, there exists a proposition $\langle v/d\rangle \in \mathcal{D}$ such that $u_v(d) > 0$ and $\langle v/d\rangle \in s_0[\![\pi]\!]$. Likewise, since $s_{0\varepsilon} := s_0 \cup \{\langle g/0\rangle\}$ and $O_\varepsilon \supseteq O$, π is applicable in $s_{0\varepsilon}$. Hence, by definition of $o_{\langle v/d\rangle} \in O_\varepsilon$, $\pi \cdot \langle o_{\langle v/d\rangle}\rangle$ is applicable in $s_{0\varepsilon}$ and $\langle g/1\rangle \in s_{0\varepsilon}[\![\pi \cdot \langle o_{\langle v/d\rangle}\rangle]\!]$, that is, $\pi \cdot \langle o_{\langle v/d\rangle}\rangle \in \mathcal{P}_\varepsilon$. In turn, if L is a landmark for Π_ε, then $\pi \cdot \langle o_{\langle v/d\rangle}\rangle$ contains an operator from L, and if $L \subseteq O$, then π contains an operator from L as well. This proves that all landmarks L for Π_ε over the operators O of Π are ε-landmarks for Π. $\square$

With Theorem 15 in hand, we can now derive ε-landmarks for Π using any method for classical planning landmark extraction, such as that employed by the LAMA planner (Richter et al., 2008) or the LM-Cut family of techniques (Helmert & Domshlak, 2009; Bonet & Helmert, 2010). However, at first glance, the discriminative power of knowing "what is needed to achieve *something* valuable" seems to be negligible when it comes to deriving effective heuristic estimates for OSP. The good news is that, in OSP, such information can be effectively exploited in a slightly different way.

Consider a schematic example of searching for an optimal plan for an OPS task Π with budget b, using *BFBB* with an admissible heuristic h. Suppose that there is only one sequence of (all unit-cost) operators, $\pi = \langle o_1, o_2, \ldots, o_{b+1} \rangle$, applicable in the initial state of Π, and that the only positive value state along π is its end-state. While clearly no value higher than zero can be achieved in Π under the given budget of b, the search will continue beyond the initial state, unless $h(s_0, \cdot)$ counts the cost of all the $b+1$ operators of π. Now, suppose that $h(s_0, \cdot)$ counts only the cost of $\{o_i, \ldots, o_{b+1}\}$ for some $i > 0$, but $\{o_1\}, \{o_2\}, \ldots, \{o_{i-1}\}$ are all discovered to be ε-landmarks for Π. Given that, suppose that we modify Π by (a) setting the cost of operators $o_1, o_2, \ldots, o_{i-1}$ to zero, and (b) *reducing the budget* to $b - i + 1$. Since all the operators $o_1, o_2, \ldots, o_{i-1}$ have to be applied anyway along any value collecting plan for Π, this modification seems to preserve the semantics of Π. At the same time, on the modified task, *BFBB* with the same heuristic h will prune the initial state and thus establish without any search that the empty plan is an optimal plan for Π. Of course, the way Π is modified in this example is as simplistic as the example itself. Yet, this example does motivate the idea of *landmark-based budget reduction* for OSP, as well as illustrates the basic idea behind the *generically sound* task modifications that we discuss next.

Definition 6 *Let* $\Pi = \langle V, s_0, u; O, c, b \rangle$ *be an OSP task,* $\mathcal{L} = \{L_1, \ldots, L_n\}$ *be a set of pairwise disjoint* ε*-landmarks for* Π*, and* $lcost$ *be an admissible landmark cost function from* $\mathcal{L}$*. The* **budget reducing compilation** *of* Π *is an OSP task* $\Pi_{\mathcal{L}} = \langle V_{\mathcal{L}}, s_{0\mathcal{L}}, u_{\mathcal{L}}; O_{\mathcal{L}}, c_{\mathcal{L}}, b_{\mathcal{L}} \rangle$ *where*

$$b_{\mathcal{L}} = b - \sum_{i=1}^{n} lcost(L_i) \tag{14}$$

and

$$\begin{aligned}
V_{\mathcal{L}} &= V \cup \{v_{L_1}, \ldots, v_{L_n}\} \\
&\qquad\qquad \text{with } dom(v_{L_i}) = \{0, 1\}, \\
s_{0\mathcal{L}} &= s_0 \cup \{\langle v_{L_1}/1 \rangle, \ldots, \langle v_{L_n}/1 \rangle\}, \\
u_{\mathcal{L}} &= u, \\
O_{\mathcal{L}} &= O \cup \bigcup_{i=1}^{n} O_{L_i} = O \cup \bigcup_{i=1}^{n} \{\overline{o} \mid o \in L_i\}, \\
&\qquad \text{with } \mathsf{pre}(\overline{o}) = \mathsf{pre}(o) \cup \{\langle v_{L_i}/1 \rangle\} \text{ and } \mathsf{eff}(\overline{o}) = \mathsf{eff}(o) \cup \{\langle v_{L_i}/0 \rangle\}, \\
c_{\mathcal{L}}(\omega) &= \begin{cases} c(o), & \omega = o \in O \\ c(o) - lcost(L_i), & \omega = \overline{o} \in O_{L_i} \end{cases}.
\end{aligned}$$

In other words, $\Pi_{\mathcal{L}}$ extends the structure of Π by

- mirroring the operators of each ε-landmark L_i with their "cheaper by $lcost(L_i)$" versions,

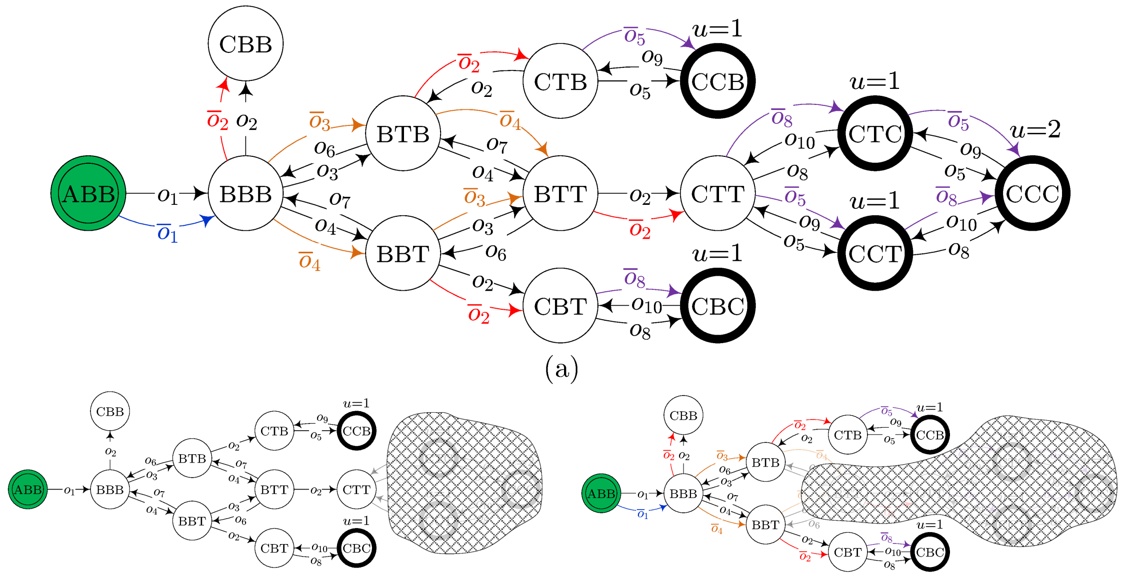


Figure 12: Illustrations for our example of the landmark-based budget reducing compilation $\Pi_{\mathcal{L}}$: (a) The structurally reachable parts of the graphical skeleton of the model induced by $\Pi_{\mathcal{L}}$, illustrated on the projection of $\Pi_{\mathcal{L}}$ on the variables of the original task Π, along with a comparison between the budget-wise reachable parts of the graphical skeletons induced by the models of the (b) original task Π and (c) the compiled task $\Pi_{\mathcal{L}}$.

- using the "disposable" propositions $\langle v_{L_1}/1\rangle, \dots, \langle v_{L_n}/1\rangle$ to ensure that at most one instance of these discounted operators for each L_i can be applied along an operator sequence from the initial state[12], and
- compensating for the discounted operators for L_i by reducing the budget by precisely $lcost(L_i)$.

For example, consider the simple OSP task Π from Figure 2 (p. 104) with cost budget $b = 4$, and assume we are provided with a set of four landmarks $\mathcal{L} = \{L_1, \dots, L_4\}$ where $L_1 = \{o_1\}$, $L_2 = \{o_2\}$, $L_3 = \{o_3, o_4\}$ and $L_4 = \{o_5, o_8\}$, and with admissible landmark cost function $lcost(L_i) = 1$ for $i \in [4]$. Compiling $(\mathcal{L}, lcost)$ into Π using the budget reducing compilation in Definition 6 results in a task $\Pi_{\mathcal{L}}$ with budget $b_{\mathcal{L}} = 0$ and $c(\overline{o}) = 0$ for all the discounted operators $\overline{o} \in \bigcup_{i=1}^{n} O_{L_i} = \{\overline{o}_1, \overline{o}_2, \overline{o}_3, \overline{o}_4, \overline{o}_5, \overline{o}_8\}$.

While the states of Π correspond to complete assignments to only three variables $V = \{t, x, y\}$, $\Pi_{\mathcal{L}}$ already has seven variables $V_{\mathcal{L}} = \{t, x, y, v_{L_1}, v_{L_2}, v_{L_3}, v_{L_4}\}$. Thus, depicting

12. Note that, while the auxiliary variable g in the ε-compilation Π_ε of Π can effectively change its value only from $\langle g/0\rangle$ to $\langle g/1\rangle$, the auxiliary variables v_{L_i} in $\Pi_{\mathcal{L}}$ change their values (only) from $\langle v_{L_i}/1\rangle$ to $\langle v_{L_i}/0\rangle$. This difference reflects the "positive" semantics that is usually associated with the value 1, aka value *true*, of the planning propositions: The semantics of a state s of $\Pi_{\mathcal{L}}$ containing the proposition $\langle v_{L_i}/1\rangle$ is that we are still allowed to apply (one of the) discounted operators associated with the landmark L_i from s onwards.

compile-and-BFBB ($\Pi = \langle V, s_0, u; O, c, b \rangle$)
  Π_ε := ε-compilation of Π
  $\mathcal{L}$:= a set of landmarks for Π_ε
  $lcost$:= admissible landmark cost function from $\mathcal{L}$
  $\Pi_{\mathcal{L}^*}$:= budget reducing compilation of $(\mathcal{L}, lcost)$ into Π
  n^* := BFBB($\Pi_{\mathcal{L}^*}$)
  return plan for Π associated with n^*

Figure 13: *BFBB* search with landmark-based budget reduction

the structurally reachable parts of the graphical skeleton $G_{M_{\Pi_\mathcal{L}}}$ is problematic. Still, to illustrate the search space of Π, in Figure 12(a) we show the (structurally reachable parts of the) graphical skeleton of the model induced by the projection of Π on the variables $\{t, x, y\}$ only. The arcs corresponding to discounted operators are colored, with each color distinguishing the landmark responsible for the respective discounted operators.

Figures 12(b) and 12(c) illustrate the effect of the budget-reducing compilation by depicting the parts of the graphical skeletons G_{M_Π} and $G_{M_{\Pi_\mathcal{L}}}$ that are actually reachable under the respective cost budgets $b = 4$ and $b_\mathcal{L} = 0$: While the states BTT and CTT are reachable from the initial state of Π under the budget allowance of 4, the states corresponding to BTT and CTT are no longer reachable in $\Pi_\mathcal{L}$, reducing the size of the search space for *BFBB*. At the same time, as formulated in Theorem 16 below, this reduction of the search space does not affect plans for Π that lead to valuable states, resulting in an effective equivalence between Π and $\Pi_\mathcal{L}$.

Theorem 16 *Let $\Pi = \langle V, s_0, u; O, c, b \rangle$ be an OSP task, $\mathcal{L}$ be a set of pairwise disjoint ε-landmarks for Π, lcost be an admissible landmark cost function from $\mathcal{L}$, and $\Pi_\mathcal{L}$ be the respective budget reducing compilation of Π. For every π for Π with $Q^b(\pi) > 0$, there is a plan $\pi_\mathcal{L}$ for $\Pi_\mathcal{L}$ with $Q^{b_\mathcal{L}}(\pi_\mathcal{L}) = Q^b(\pi)$, and vice versa.*

The proof of Theorem 16 appears in Appendix A, p. 149. The budget reducing OSP-to-OSP compilation in Definition 6 is clearly polynomial time. The compile-and-BFBB procedure, depicted in Figure 13,

(1) generates an ε-compilation Π_ε of Π;

(2) uses off-the-shelf tools for classical planning to generate a set of landmarks $\mathcal{L}$ for Π_ε and an admissible landmark cost function *lcost*; and

(3) compiles $(\mathcal{L}, lcost)$ into Π, obtaining an OSP task $\Pi_\mathcal{L}$.

The optimal solution for $\Pi_\mathcal{L}$ (and thus for Π) is then searched for using a search algorithm for optimal OSP such as *BFBB*.

Before we proceed to consider more general sets of landmarks, a few comments concerning the setup of Theorem 16 are in order. First, if the reduced budget $b_\mathcal{L}$ turns out to be lower than the cost of the cheapest action applicable in the initial state, then obviously no search is needed, and the empty plan can be reported as optimal right away. Second, zero-cost landmarks are useless in our compilation as much as they are useless in deriving landmark heuristics for optimal planning. Hence, *lcost* in what follows is assumed to be strictly positive. Third, having both o and $\overline{o}$ applicable at a state of Π_ε brings no benefits yet adds branching to the search. Hence, in our implementation, for each landmark $L_i \in \mathcal{L}$ and each operator $o \in L_i$, the precondition of the regular operators o in $O_\mathcal{L}$ is extended with $\{\langle v_{L_i}/0\rangle\}$. It is not hard to verify that this extension preserves the correctness of $\Pi_\mathcal{L}$ in terms of Theorem 16. Finally, if the value of the initial state is not zero, that is, the empty plan has some positive value, then ε-compilation Π_ε of Π will have no positive cost landmarks at all. However, this can easily be fixed by considering as "valuable" only propositions $\langle v/d\rangle$ such that both $u_v(d) > 0$ and $\langle v/d\rangle \notin s_0$. We ignore for the time being the problem of non-zero-value initial states (and assume that $Q^b(\epsilon) = 0$), but return to it later for a more systematic discussion.

5.3 Non-Disjoint ε-Landmarks

While the budget reducing compilation $\Pi_\mathcal{L}$ above is sound for pairwise disjoint landmarks, this is not so for more general sets of ε-landmarks. For example, consider a planning task Π in which, for some operator o, we have $c(o) = b$, $Q^b(\langle o\rangle) > 0$, and $Q^b(\pi) = 0$ for all other operator sequences $\pi \neq \langle o\rangle$. That is, a value greater than zero is achievable in Π, but only via the operator o. Suppose now that our set of ε-landmarks for Π is $\mathcal{L} = \{L_1, \dots, L_n\}$, $n > 1$, with $lcost(L_i) > 0$ for all $i \in [n]$, and that all of these ε-landmarks contain o. In this case, while the budget in $\Pi_\mathcal{L}$ is $b_\mathcal{L} = b - \sum_{i=1}^n lcost(L_i)$, the cost of the cheapest replica $\overline{o}$ of o, that is, the cost of the cheapest operator sequence achieving a non-zero value in Π, is

$$c(o) - \max_{i=1}^{n} lcost(L_i) = b - \max_{i=1}^{n} lcost(L_i) > b - \sum_{i=1}^{n} lcost(L_i) = b_\mathcal{L}.$$

Hence, no state with positive value will be reachable from $s_{0\mathcal{L}}$ in $\Pi_\mathcal{L}$, and thus Π and $\Pi_\mathcal{L}$ are not "value equivalent" in the sense of Theorem 16.

This example shows why compiling non-disjoint ε-landmarks into Π independently is not sound. In principle, it can be made sound as follows. Let $\Pi = \langle V, s_0, u; O, c, b\rangle$ be an OSP task, let $\mathcal{L} = \{L_1, \dots, L_n\}$ be a set of ε-landmarks for Π, and let *lcost* be an admissible landmark cost function from $\mathcal{L}$. All the components in $\Pi_\mathcal{L} = \langle V_\mathcal{L}, s_{0\mathcal{L}}, u_\mathcal{L}; O_\mathcal{L}, c_\mathcal{L}, b_\mathcal{L}\rangle$ are still defined as in Definition 6, except for the operator sets $O_{L_1}, \dots, O_{L_n}$. The latter are now constructed not independently of each other, but sequentially, with the content of O_{L_i} depending on the content of all O_{L_j}, $j < i$. The ordering in which the sets O_{L_i} are constructed can be arbitrary.

For each operator $o \in O$ and each $1 \leq i \leq n$, let $\overline{O}_{o;i}$ denote the set of all "cost discounted" representatives of o introduced during the construction of $O_{L_1}, \dots, O_{L_i}$. For $1 \leq i \leq n$, if for some operator $o \in \bigcup_{o' \in L_i} \overline{O}_{o';i-1}$ we have $c_\mathcal{L}(o) = 0$, then $O_{L_i} := \emptyset$. Otherwise, O_{L_i} contains an operator $\overline{o}$ for each operator

$$o \in L_i \cup \bigcup_{o' \in L_i} \overline{O}_{o';i-1}, \tag{15}$$

with $\overline{o}$ being defined very similarly to Definition 6 as:

$$\begin{aligned}
\mathsf{pre}(\overline{o}) &= \mathsf{pre}(o) \cup \{\langle v_{L_i}/1\rangle\},\\
\mathsf{eff}(\overline{o}) &= \mathsf{eff}(o) \cup \{\langle v_{L_i}/0\rangle\},\\
c_{\mathcal{L}}(\overline{o}) &= \begin{cases} c(o) - lcost(L_i), & o \in L_i,\\ c_{\mathcal{L}}(o) - lcost(L_i), & o \in \bigcup_{o' \in L_i} \overline{O}_{o';i-1} \end{cases}.
\end{aligned} \tag{16}$$

The compilation extended this way is sound for arbitrary sets of ε-landmarks, and on pairwise disjoint landmarks it reduces to the basic compilation used in Theorem 16. In general, however, this extended compilation is no longer polynomial in the size of the explicit representation of Π because

$$|\overline{O}_{o;i}| = 2^{|\{L_j | j \leq i, o \in L_j\}|}.$$

For example, let $\mathcal{L} = \{L_1, L_2, L_3\}$, $L_1 = \{a, b\}$, $L_2 = \{a, c\}$, $L_3 = \{a, d\}$. Generation of $O_{L_1} := \{\overline{a}_1, \overline{b}_1\}$ effectively follows Definition 6, but for O_{L_2}, the base set of operators as in Eq. 15 is already $\{a, c, \overline{a}_1\}$. Thus, $O_{L_2} := \{\overline{a}_2, \overline{c}_1, \overline{a}_3\}$, where, for $i \in \{2, 3\}$ and denoting a by $\overline{a}_0$, $\overline{a}_i$ is derived according to Eq. 16 from $\overline{a}_{i-2}$. Consequently, the base set of operators for O_{L_3} is $\{a, d, \overline{a}_1, \overline{a}_2, \overline{a}_3\}$, resulting in $O_{L_3} = \{\overline{a}_4, \overline{d}_1, \overline{a}_5, \overline{a}_6, \overline{a}_7\}$, where, for $i \in \{4, 5, 6, 7\}$, $\overline{a}_i$ is derived from $\overline{a}_{i-4}$. In sum, $\Pi_{\mathcal{L}}$ ends up with $8 = 2^{|\mathcal{L}|}$ representatives of the operator a.

Since non-disjoint landmarks can bring more information, and they are typical to outputs of standard techniques for landmark extraction in classical planning, we now present a different, slightly more involved, compilation that is both polynomial and sound for arbitrary sets of ε-landmarks.

Definition 7 *Let $\Pi = \langle V, s_0, u; O, c, b\rangle$ be an OSP task, $\mathcal{L} = \{L_1, \ldots, L_n\}$ be a set of pairwise disjoint ε-landmarks for Π, and lcost be an admissible landmark cost function from $\mathcal{L}$. For each operator o, let $\mathcal{L}(o)$ denote the set of all landmarks in $\mathcal{L}$ that contain o. Then, the* **generalized budget reducing compilation** *of Π is an OSP task $\Pi_{\mathcal{L}^*} = \langle V_{\mathcal{L}^*}, s_{0\mathcal{L}^*}, u_{\mathcal{L}^*}; O_{\mathcal{L}^*}, c_{\mathcal{L}^*}, b_{\mathcal{L}^*}\rangle$ where*

$$\begin{aligned}
b_{\mathcal{L}^*} &= b - \sum_{i=1}^{n} lcost(L_i),\\
V_{\mathcal{L}^*} &= V \cup \{v_{L_1}, \ldots, v_{L_n}\}\\
&\qquad \text{with } dom(v_{L_i}) = \{0, 1\},\\
s_{0\mathcal{L}^*} &= s_0 \cup \{\langle v_{L_1}/1\rangle, \ldots, \langle v_{L_n}/1\rangle\},\\
u_{\mathcal{L}^*} &= u,\\
O_{\mathcal{L}^*} &= O \cup \{\overline{o} \mid o \in \cup_{L \in \mathcal{L}} L\} \cup \{get(L) \mid L \in \mathcal{L}\}
\end{aligned}$$

with

$$\begin{aligned}
\mathsf{pre}(\overline{o}) &= \mathsf{pre}(o) \cup \{\langle v_L/1\rangle \mid L \in \mathcal{L}(o)\},\\
\mathsf{eff}(\overline{o}) &= \mathsf{eff}(o) \cup \{\langle v_L/0\rangle \mid L \in \mathcal{L}(o)\},
\end{aligned} \tag{17}$$

and

$$\begin{aligned}
\mathsf{pre}(get(L)) &= \{\langle v_L/0\rangle\},\\
\mathsf{eff}(get(L)) &= \{\langle v_L/1\rangle\},
\end{aligned} \tag{18}$$

and

$$c_{\mathcal{L}^*}(\omega) = \begin{cases} c(o), & \omega = o \in O \\ c(o) - \sum_{L \in \mathcal{L}(o)} lcost(L), & \omega = \overline{o} \\ lcost(L), & \omega = get(L) \end{cases}. \tag{19}$$

To illustrate this compilation, we will let $\mathcal{L} = \{L_1, L_2, L_3\}$,

$$\begin{aligned} L_1 &= \{a, b\}, \\ L_2 &= \{b, c\}, \\ L_3 &= \{a, c\}, \end{aligned}$$

with all operators having the cost of 2, and let

$$lcost(L_1) = lcost(L_2) = lcost(L_3) = 1.$$

In $\Pi_{\mathcal{L}^*}$, we have $V_{\mathcal{L}^*} = V \cup \{v_{L_1}, v_{L_2}, v_{L_3}\}$ and

$$O_{\mathcal{L}^*} = O \cup \{\overline{a}, \overline{b}, \overline{c}, get(L_1), get(L_2), get(L_3)\},$$

with, e.g.,

$$\begin{aligned} \mathsf{pre}(\overline{a}) &= \mathsf{pre}(a) \cup \{\langle v_{L_1}/1\rangle, \langle v_{L_3}/1\rangle\}, \\ \mathsf{eff}(\overline{a}) &= \mathsf{eff}(a) \cup \{\langle v_{L_1}/0\rangle, \langle v_{L_3}/0\rangle\}, \\ c_{\mathcal{L}^*}(\overline{a}) &= 0, \end{aligned}$$

and, for $get(L_1)$,

$$\begin{aligned} \mathsf{pre}(get(L_1)) &= \{\langle v_L/0\rangle\}, \\ \mathsf{eff}(get(L_1)) &= \{\langle v_L/1\rangle\}, \\ c_{\mathcal{L}^*}(get(L_1)) &= 1. \end{aligned}$$

The intuition behind the compilation in Definition 7 is as follows. By Eq. 19, applying a discounted operator $\overline{o}$ saves the total cost of all landmarks containing o. Therefore,

- $\overline{o}$ can be executed only at states s in which all the corresponding control propositions $\{\langle v_L/1\rangle \mid L \in \mathcal{L}(o)\}$ hold, indicating that the cost of no landmark in $\mathcal{L}(o)$ has already been saved before reaching s, and
- to avoid double savings around $\mathcal{L}(o)$, applying $\overline{o}$ in s turns off all these control propositions in $s[\![\overline{o}]\!]$.

However, considering the example above, suppose that the optimal plan π for the original task contains an instance of operator a, followed by an instance of operator b, and no instance of operator c. Applying $\overline{a}$ instead of a would block us from applying $\overline{b}$ instead of b, and thus the value of the optimal plan in the compilation can be lower than $Q^b(\pi)$. The rescue here comes from the $get(L)$ actions that allow for *selective* "spending" of the *individual* landmark costs $lcost(L)$. In our example, while applying $\overline{a}$ in s saves the cost of the landmarks L_1 and L_3, applying $get(L_1)$ will then spend $lcost(L_1)$ and safely set the control proposition $\langle v_{L_1}/1 \rangle$. In turn, this will enable $\overline{b}$ to be applied at the next steps, and applying $\overline{b}$ will then save the cost of L_2 and "re-save" the cost of L_1. This way, the compilation leads to equivalence between Π and $\Pi_{\mathcal{L}^*}$, formulated in Theorem 17 below and proven in Appendix A, p. 149.

Theorem 17 *Let $\Pi = \langle V, s_0, u; O, c, b \rangle$ be an OSP task, let $\mathcal{L} = \{L_1, \ldots, L_n\}$ be a set of ε-landmarks for Π, let lcost be an admissible landmark cost function from $\mathcal{L}$, and let $\Pi_{\mathcal{L}^*}$ be the (generalized) budget reducing compilation of Π. For every π for Π with $Q^b(\pi) > 0$, there is a plan $\pi_{\mathcal{L}^*}$ for $\Pi_{\mathcal{L}^*}$ with $Q^{b_{\mathcal{L}^*}}(\pi_{\mathcal{L}^*}) = Q^b(\pi)$, and vice versa.*

5.4 ε-Landmarks & Incremental *BFBB*

As we discussed earlier, if the value of the initial state is not zero, then the empty plan has some positive value, and thus the ε-compilation Π_ε of Π as in Definition 5 will have no landmarks with positive cost. In passing we noted that this small problem can be remedied by considering as "valuable" only facts $\langle v/d \rangle$ such that both $u_v(d) > 0$ and $\langle v/d \rangle \notin s_0$. We now consider this aspect of OSP more closely, and show how the discovery of ε-landmarks and the incremental revelation of plans by *BFBB* can be combined in a mutually stratifying way.

Let $\Pi = \langle V, s_0, u; O, c, b \rangle$ be the OSP task of interest, and suppose we are *given* a set of plans $\pi_1, \ldots, \pi_n$ for Π. If so, then we are no longer interested in searching for plans that "achieve something," but in searching for plans that achieve *something beyond what* $\pi_1, \ldots, \pi_n$ *already achieve*. Specifically, let $s_i = s_0[\![\pi_i]\!]$ be the end-state of π_i, and for any set of propositions $s \subseteq \mathcal{D}$, let $\mathsf{goods}(s) \subseteq s$ be the set of all propositions $\langle v/d \rangle \in s$ such that $u_v(d) > 0$. If a new plan π with end-state s achieves something beyond what $\pi_1, \ldots, \pi_n$ already achieve, then, for *all* $1 \leq i \leq n$,

$$\mathsf{goods}(s) \setminus \mathsf{goods}(s_i) \neq \emptyset.$$

We now put this observation to work.

Definition 8 *Given an OSP task $\Pi = \langle V, s_0, u; O, c, b \rangle$ and a set of reference states $S_{ref} = \{s_1, \ldots, s_n\}$ of Π, the* $(\varepsilon, S_{\mathbf{ref}})$**-compilation** *of Π is a classical planning task $\Pi_{(\varepsilon, S_{ref})} = \langle V_\varepsilon, s_{0\varepsilon}, G_\varepsilon; O_\varepsilon, c_\varepsilon \rangle$ with*

$$\begin{aligned}
V_\varepsilon &= V \cup \{x_1, \ldots, x_n, \mathit{search}, \mathit{collect}\}, \\
&\quad \text{with } \mathit{dom}(x_i) = \mathit{dom}(\mathit{search}) = \mathit{dom}(\mathit{collect}) = \{0, 1\}, \\
s_{0\varepsilon} &= s_0 \cup \{\langle \mathit{search}/1 \rangle, \langle \mathit{collect}/0 \rangle, \langle x_1/0 \rangle, \ldots, \langle x_n/0 \rangle\}, \\
G_\varepsilon &= \{\langle x_1/1 \rangle, \ldots, \langle x_n/1 \rangle\}, \\
O_\varepsilon &= \overline{O} \cup \bigcup_{i=1}^{n} O_i \cup \{\mathit{finish}\},
\end{aligned}$$

where

- $\overline{O} = \{\overline{o} \mid o \in O\}$,

$$\begin{aligned}\mathsf{pre}(\overline{o}) &= \mathsf{pre}(o) \cup \{\langle search/1\rangle\},\\ \mathsf{eff}(\overline{o}) &= \mathsf{eff}(o),\\ c_\varepsilon(\overline{o}) &= c(o).\end{aligned}$$

-

$$\begin{aligned}\mathsf{pre}(finish) &= \emptyset,\\ \mathsf{eff}(finish) &= \{\langle collect/1\rangle, \langle search/0\rangle\},\\ c_\varepsilon(finish) &= 0.\end{aligned}$$

- $O_i = \{o_{i,g} \mid s_i \in S_{ref}, g \in \mathsf{goods}(\mathcal{D}) \setminus s_i\}$,

$$\begin{aligned}\mathsf{pre}(o_{i,g}) &= \{g, \langle collect/1\rangle\},\\ \mathsf{eff}(o_{i,g}) &= \{\langle x_i/1\rangle\},\\ c_\varepsilon(o_{i,g}) &= 0.\end{aligned}$$

Note that

- the goal G_ε cannot be achieved without applying the $finish$ operator;
- the "regular" operators $\overline{o}$ can be applied only before $finish$; and
- the subgoal achieving operators $o_{i,g}$ can be applied only after $finish$.

This way, the first part of any plan for $\Pi_{(\varepsilon,S_{\text{ref}})}$ determines a plan for Π, and the second part "verifies" that the end-state of that plan achieves a subset of value-carrying propositions $\mathsf{goods}(\mathcal{D})$ that is included in no state from S_{ref}.[13]

Theorem 18 *Let $\Pi = \langle V, s_0, u; O, c, b\rangle$ be an OSP task, $S_{ref} = \{s_1, \dots, s_n\}$ be a subset of Π's states, and L be a landmark for $\Pi_{(\varepsilon,S_{ref})}$ such that $L \subseteq \overline{O}$. For any plan π for Π such that $\mathsf{goods}(s_0[\![\pi]\!]) \setminus \mathsf{goods}(s_i) \neq \emptyset$ for all $s_i \in S_{ref}$, π contains an instance of at least one operator from $L' = \{o \mid \overline{o} \in L\}$.*

Proof: Assume to the contrary that there exists a plan $\pi = \langle o_1, \dots, o_k\rangle$ for Π such that $\mathsf{goods}(s_0[\![\pi]\!]) \setminus \mathsf{goods}(s_i) \neq \emptyset$ for all $s_i \in S_{\text{ref}}$, and yet $\pi \cap L' = \emptyset$. Let $\{g_1, \dots, g_n\}$ be an arbitrary set of propositions from $\mathsf{goods}(s_0[\![\pi]\!]) \setminus \mathsf{goods}(s_1), \dots, \mathsf{goods}(s_0[\![\pi]\!]) \setminus \mathsf{goods}(s_n)$, respectively. By the construction of $\Pi_{(\varepsilon,S_{\text{ref}})}$, it is immediate that

$$\pi_{(\varepsilon,S_{\text{ref}})} = \langle \overline{o}_1, \dots, \overline{o}_k, finish, o_{1,g_1}, \dots, o_{n,g_n}\rangle$$

is a plan for $\Pi_{(\varepsilon,S_{\text{ref}})}$ and, by our assumption about π and L', it holds that $\pi_{(\varepsilon,S_{\text{ref}})} \cap L = \emptyset$. This, however, contradicts that L is a landmark for $\Pi_{(\varepsilon,S_{\text{ref}})}$. $\square$

13. This "solve & verify" technique appears to be helpful in many planning formalism compilations; see, e.g., the work of Keyder and Geffner (2009).

```
inc-compile-and-BFBB (Π = ⟨V, O; s_0, c, u, b⟩)
  initialize global variables:
      n* := s_0            // best solution so far
      S_ref := {s_0}       // current reference states
  loop:
    Π_(ε,S_ref) = (ε, S_ref)-compilation of Π
    L := a set of landmarks for Π_(ε,S_ref)
    lcost := admissible landmark cost function from L
    Π_L* := budget reducing compilation of (L, lcost) into Π
    if inc-BFBB(Π_L*, S_ref, n*) = done:
      return plan for Π associated with n*

inc-BFBB (Π, S_ref, n*)
  open := new max-heap ordered by f(n) = h(s⟨n⟩, b − g(n))
  open.insert(make-root-node(s_0))
  closed:= ∅
  best-cost:= 0
  while not open.empty()
    n := open.pop-max()
    if f(n) ≤ u(s⟨n*⟩): break
    if u(s⟨n⟩) > u(s⟨n*⟩): update n* := n
    if goods(s⟨n⟩) ⊈ goods(s') for all s' ∈ S_ref:
      S_ref := S_ref ∪ {s⟨n⟩}
      if termination criterion: return updated
    // the rest is similar to BFBB in Figure 3
    if s⟨n⟩ ∉ closed or g(n) < best-cost(s⟨n⟩):
      closed:= closed ∪ {s⟨n⟩}
      best-cost(s⟨n⟩) := g(n)
      foreach o ∈ O(s⟨n⟩):
              n' := make-node(s⟨n⟩⟦o⟧, n)
              if g(n') > b or f(n') ≤ u(s⟨n*⟩): continue
              open.insert(n')
  return done
```

Figure 14: Iterative *BFBB* with landmark enhancement

Theorem 18 allows us to define an iterative version of *BFBB*, inc-compile-and-BFBB, depicted in Figure 14. The successive iterations of inc-compile-and-BFBB correspond to running the regular *BFBB* on successively more informed $(\varepsilon, S_{\text{ref}})$-compilations of Π, with the states discovered at iteration i making the $(\varepsilon, S_{\text{ref}})$-compilation used at iteration $i+1$ more informed.

inc-compile-and-BFBB maintains as a pair of global variables: a set of reference states S_{ref} and the best solution so far n^*. At each iteration of the loop, a modified version of *BFBB*, inc-BFBB, is called with an $(\varepsilon, S_{\text{ref}})$-compilation of Π, created on the basis of the *current* S_{ref}. The reference set S_{ref} is then extended by inc-BFBB with all the non-redundant value-carrying states discovered during the search, and n^* is updated if the search discovers nodes of higher value.

If and when the OPEN list becomes empty or the node n selected from the list promises less than the lower bound, inc-BFBB returns an indicator, *done*, that the best solution n^* found so far, across the iterations of inc-compile-and-BFBB, is optimal. In that case, inc-compile-and-BFBB leaves its loop and extracts that optimal plan from n^*. However, inc-BFBB may also terminate in a different way, if a certain complementary termination criterion is satisfied. The latter criterion comes to assess whether the updates to S_{ref} performed in the current session of inc-BFBB warrant updating the $(\varepsilon, S_{\text{ref}})$-compilation and restarting the search. If terminated this way, inc-BFBB returns a respective indicator, and inc-compile-and-BFBB goes into another iteration of its loop, with the *updated* S_{ref} and n^*. We note that, while the optimality of the algorithm holds for *any* such termination condition, the latter should greatly affect the runtime efficiency of the algorithm.

Theorem 19 *The* inc-compile-and-BFBB *search algorithm is sound and complete for optimal OSP.*

Proof: First, for any complementary termination criterion employed by the inc-BFBB procedure, inc-compile-and-BFBB is guaranteed to terminate. This is because that complementary termination criterion is checked in inc-BFBB only after a proper expansion of the global reference set S_{ref}, and thus the number of calls to inc-BFBB by inc-compile-and-BFBB is upper-bounded by $|S|$.

It terms of search, inc-BFBB is no different from the regular *BFBB* procedure. In turn, by Theorem 18, the additional pruning power of the budget-reducing compilation with reference states S_{ref} affects only search nodes n such that $u(s\langle n\rangle) < \max_{s \in S_{\text{ref}}} u(s)$. Note also that, each time the best solution so far n^* is updated in inc-BFBB, it is necessarily added to S_{ref} (since $\mathsf{goods}(s\langle n^*\rangle)$ of the new n^* can be included in no $\mathsf{goods}(s)$ for $s \in S_{\text{ref}}$). Thus, optimal solutions cannot be pruned out by inc-BFBB and the overall search by inc-compile-and-BFBB is therefore sound. □

5.5 Empirical Evaluation

To evaluate the merits of the landmark-based budget reducing compilation, we have extended our prototype OSP solver from Section 3 with the following components:

- $(\varepsilon, S_{\text{ref}})$-compilation of OSP tasks Π for arbitrary sets of reference states S_{ref};
- generation of disjunctive action landmarks for $(\varepsilon, S_{\text{ref}})$-compilations using the LM-Cut procedure (Helmert & Domshlak, 2009) of Fast Downward; and
- the incremental *BFBB* procedure inc-compile-and-BFBB as in Figure 14, with the search termination criterion being satisfied (only) if the examined node n improves over the current value lower bound, i.e., n becomes the new best-so-far node n^*.

After some preliminary evaluation, we also added two optimality preserving enhancements to the search. Because the auxiliary variables of our compilations increase the dimensionality of the problem, and this is known to negatively affect the quality of the abstraction

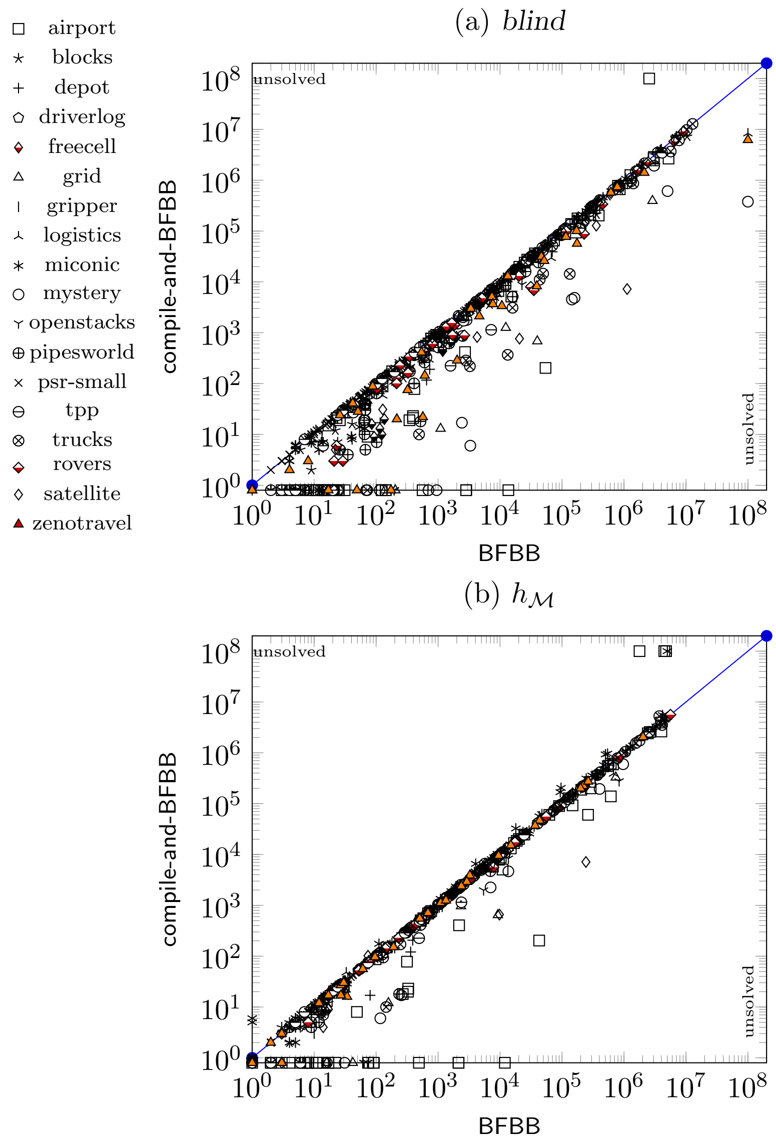


Figure 15: Comparative view of empirical results in terms of expanded nodes, for *BFBB* vs. compile-and-BFBB, with (a) blind and (b) abstraction $h_{\mathcal{M}}$ heuristics

heuristics (Domshlak et al., 2012), we first devised the projections with respect to the *original* OSP problem Π, and the open list was ordered as if the search is done on the original problem, that is, by

$$h\left(s\langle n\rangle^{\downarrow V},\ b - g(n) + \sum_{v_L \notin s\langle n\rangle} lcost(L)\right),$$

where $s^{\downarrow V}$ is the projection of the $\Pi_{\mathcal{L}^*}$'s state s on the variables of the original OSP task Π. This change in heuristic evaluation is sound, as Theorem 17 in particular implies that any admissible heuristic for Π is also an admissible heuristic for $\Pi_{\mathcal{L}^*}$, and vice versa. Second, when a new node n is generated, we check whether

$$g(n) + \sum_{L:\langle v_L/0\rangle \in s\langle n\rangle} lcost(L) \;\geq\; g(n') + \sum_{L:\langle v_L/0\rangle \in s\langle n'\rangle} lcost(L),$$

for some previously generated node n' that corresponds to the same state of the original problem Π, that is, $s\langle n'\rangle^{\downarrow V} = s\langle n\rangle^{\downarrow V}$. If so, then n is pruned right away. Optimality preservation of this enhancement is established in Lemma 20 and proven in Appendix A, p. 151.

Lemma 20 *Let Π be an OSP task, $\Pi_{(\varepsilon,S_{ref})}$ be a (ε, S_{ref})-compilation of Π, $\mathcal{L}$ be a set of landmarks for $\Pi_{(\varepsilon,S_{ref})}$, lcost be an admissible landmark cost function for $\mathcal{L}$, and $\Pi_{\mathcal{L}^*}$ be the respective budget reducing compilation of $(\mathcal{L}, lcost)$ into Π. Let π_1 and π_2 be a pair of plans for $\Pi_{\mathcal{L}^*}$ with end-states s_1 and s_2, respectively, such that $s_1^{\downarrow V} = s_2^{\downarrow V}$ and*

$$c_{\mathcal{L}^*}(\pi_1) + \sum_{L:\langle v_L/0\rangle \in s_1} lcost(L) \;\geq\; c_{\mathcal{L}^*}(\pi_2) + \sum_{L:\langle v_L/0\rangle \in s_2} lcost(L). \tag{20}$$

Then, for any plan π_1' that extends π_1, there exists a plan π_2' that extends π_2 such that $Q^{b_{\mathcal{L}^*}}(\pi_2') = Q^{b_{\mathcal{L}^*}}(\pi_1')$.

Our evaluation included the regular *BFBB* planning for Π, solving Π using landmark-based compilation via compile-and-BFBB, and the simple setting of inc-compile-and-BFBB described above. All three approaches were evaluated under the blind heuristic and the additive abstraction heuristic $h_{\mathcal{M}}$ described in Section 3. Figures 15-17 depict the results of our evaluation in terms of expanded nodes. Similarly to the experiment reported in Section 3, each task was approached under four different budgets, corresponding to 25%, 50%, 75%, and 100% of the minimal cost needed to achieve all the goals in the task, and each run was restricted to 10 minutes. Figures 15a and 15b compare the number of expanded nodes of *BFBB* and compile-and-BFBB across the four levels of cost budget, under blind (a) and abstraction $h_{\mathcal{M}}$ (b) heuristics. Figures 16a and 16b provide a similar comparison between *BFBB* and inc-compile-and-BFBB. Figures 17a and 17b do the same for compile-and-BFBB and inc-compile-and-BFBB.[14] Figures 22-25 and Figures 26-29 in Appendix B provide a more detailed view of the results in Figures 15 and 16, respectively, by breaking them down into different levels of cost budget.

As Figure 8 shows, the results were very satisfactory. With no informative heuristic guidance at all, the number of nodes expanded by compile-and-BFBB was typically much lower than the number of nodes expanded by *BFBB*, with the difference reaching three orders of magnitude more than once. Of the 760 task/budget pairs behind Figure 8a, 81 pairs were solved by compile-and-BFBB with no search at all (by proving that no plan can achieve value higher than that of the initial state), while, unsurprisingly, only 4 of these tasks were solved with no search by *BFBB*.

14. We do not present here a detailed comparison in terms of the running times, but the per-node CPU time overhead due to landmark-based budget reduction was $\leq 10\%$. Some technical difficulties with our implementation of inc-compile-and-BFBB led us to limit our comparison of it in each graph only to tasks solved by both methods.

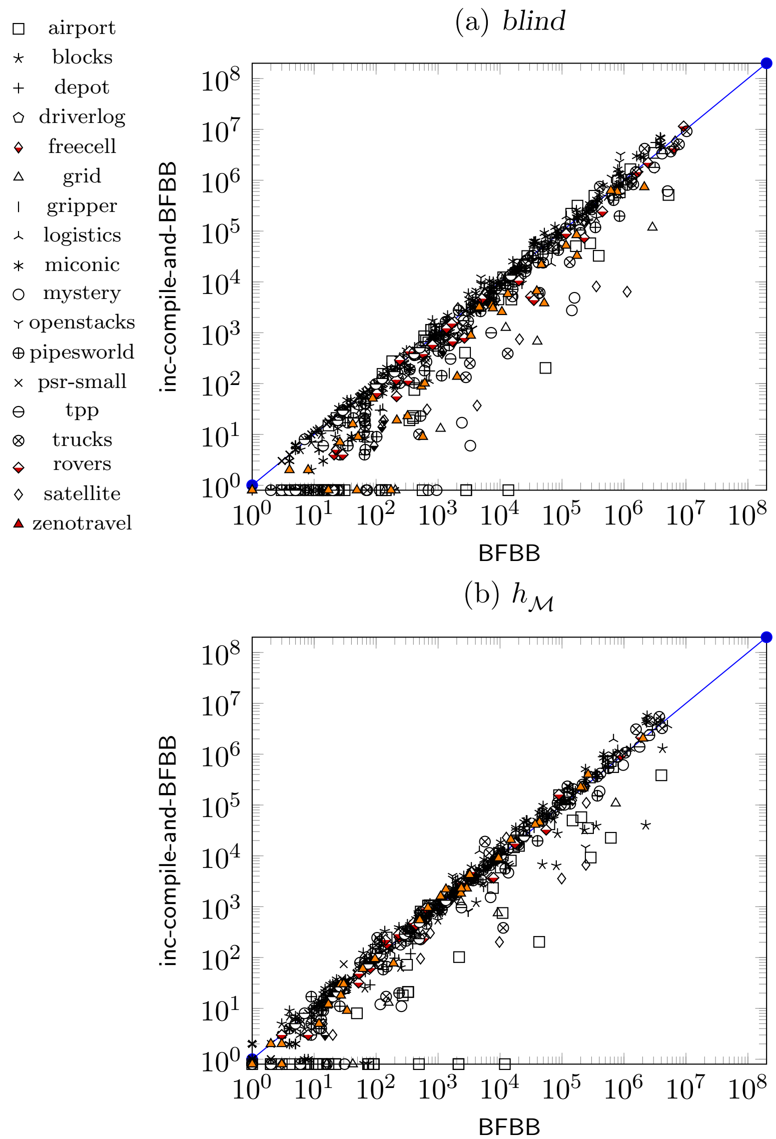


Figure 16: Comparative view of empirical results in terms of expanded nodes, for *BFBB* vs. inc-compile-and-BFBB, with (a) blind and (b) abstraction $h_{\mathcal{M}}$ heuristics

As expected, the impact of the landmark-based budget reduction is lower when the search is equipped with a meaningful heuristic (Figure 15b). Nonetheless, even with our abstraction heuristic in hand, the number of nodes expanded by compile-and-BFBB was often substantially lower than the number of nodes expanded by *BFBB*. Here, *BFBB* and compile-and-BFBB solved with no search 39 and 85 task/budget pairs, respectively. Finally, despite the rather ad hoc setting of our incremental inc-compile-and-BFBB procedure, switching from compile-and-BFBB to inc-compile-and-BFBB was typically beneficial. Obvi-

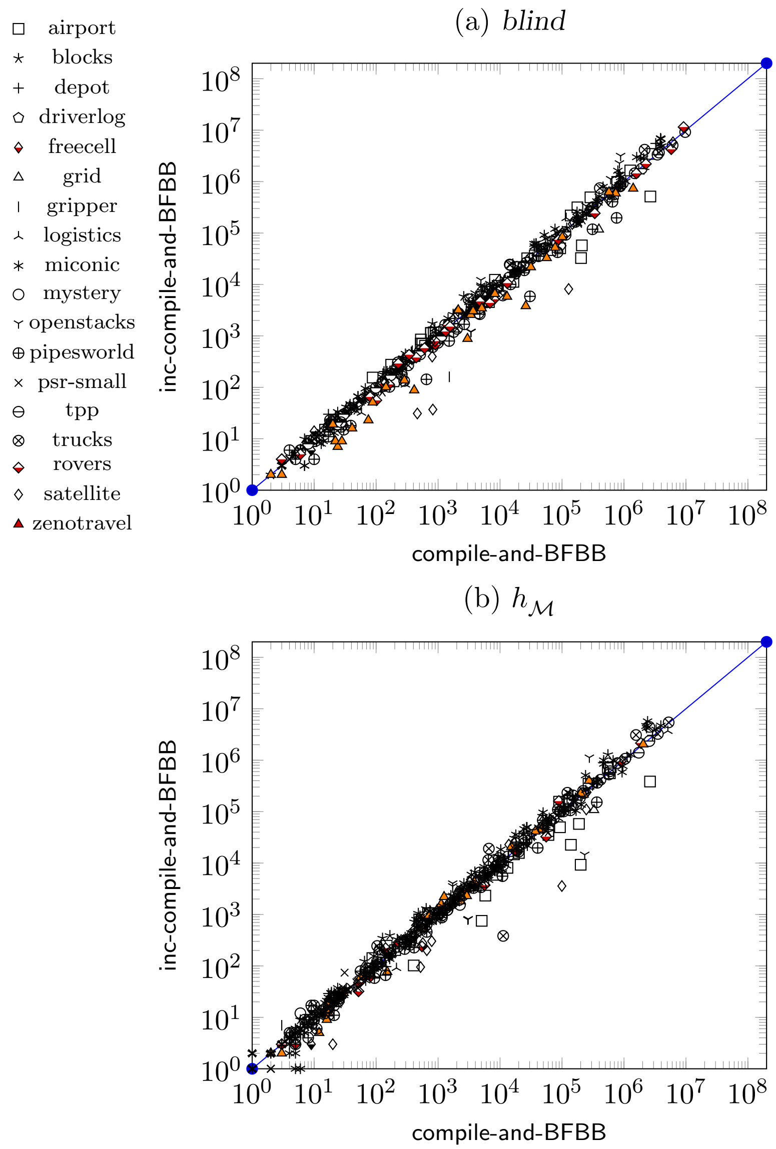


Figure 17: Comparative view of empirical results in terms of expanded nodes, for compile-and-BFBB vs. inc-compile-and-BFBB, with (a) blind and (b) abstraction $h_{\mathcal{M}}$ heuristics

ously, much deeper investigation and development of inc-compile-and-BFBB is still required, especially in the context of the choice of the iteration termination criterion.

6. Summary and Future Work

Deterministic oversubscription planning captures the computational core of one of the most important setups of automated action selection, and yet, despite the apparent importance of this problem, it has not been sufficiently investigated. In this work, we progressed towards translating the spectacular advances in classical deterministic planning to deterministic OSP. Tracing the key sources of progress in classical planning, we identified a severe lack of effective approximations for OSP, and worked towards bridging this gap.

Our focus was on two classes of approximation techniques that underly most state-of-the-art optimal heuristic-search solvers for classical planning: state-space abstractions and goal-reachability landmarks. First, we defined the notion of additive abstractions for OSP, studied the complexity of deriving effective abstractions from a rich space of hypotheses, and revealed some substantial, empirically relevant islands of tractability of this abstraction discovery problem. Next, we showed how standard goal-reachability landmarks of certain classical planning tasks can be compiled into the OSP task of interest, resulting in an equivalent OSP task with a lower cost allowance, and thus with a sometimes dramatically smaller search space.

All the techniques proposed here satisfy the properties required by the efficient search algorithms for optimal OSP. However, we believe that these techniques, and especially landmark-based budget reducing compilations, should be as beneficial in satisficing OSP as in optimal OSP, in particular because the difference between optimal and satisficing planning appears to be much smaller in OSP than in classical deterministic planning.

Many interesting questions remain open for future work, and the prospects for further developments in oversubscription planning now appear quite promising. Within the specific context of our work, the two most interesting research directions are (1) optimization of value partitions given cost partitions, that is, optimizing abstraction discovery in $\mathbf{A}_p(\mathbf{c}, -, -)$, and (2) thoroughly investigating the interleaved landmark discovery and search for OSP introduced in Section 5.4. In a broader context, we propose, as well, additional candidates for future research:

- Following the work of Katz and Domshlak (2010a) on *implicit abstractions* for classical planning, the computational merits of implicit abstractions for OSP should be investigated. This will inevitably give us a better understanding of the computational tractability boundaries of deterministic OSP.

- The basic model of deterministic planning in Section 2.1 was used to provide a unifying comparative view of the basic models of classical, cost-bounded, net-benefit, and oversubscription planning. One practically motivated extension of this model is to lift action costs to vectors of action costs. Such a variant of cost-bounded planning has already been investigated (Nakhost et al., 2012), and it is only natural to examine this extension in the context of OSP.

  Unfortunately, our results on abstractions do not seem to extend directly to vectors of costs: At the level of the planning model, adding cost measures shifts problem solving from polynomial time shortest path(s) problems to NP-hard restricted shortest path(s) problems (Handler & Zang, 1980). Nonetheless, like the Knapsack problem, the restricted shortest path problem can be solved in pseudo-polynomial time (Desrochers & Soumis, 1988), and thus some extension of our results to vectors of costs might still be achievable.

  At the same time, the machinery of landmark-based budget reducing compilations for OSP straightforwardly extends to vectors of costs and budgets. Hence, even if no quality heuristic for OSP with multiple cost measures is available, the blind search can still be stratified with information coming from problem landmarks.

- While the pruning mechanism of *BFBB* must rely on admissible, upper-bounding heuristic estimates, no special properties are required from a heuristic used to guide the search choices of *BFBB*. Thus, developing informative yet not necessarily admissible heuristics for OSP is clearly of interest.

Acknowledgments

This work was partially supported by the EOARD grant FA8655-12-1-2096, and the ISF grant 1045/12.

Appendix A. Proofs

Theorem 2 *Given an OSP task* $\Pi = \langle V, s_0, u; O, c, b\rangle$ *and a homomorphic abstraction skeleton* $\mathcal{AS} = \{(G_1, \alpha_1), \ldots, (G_k, \alpha_k)\}$ *of* M_Π*,*

(1) for each cost partition $\mathbf{c} \in \mathbf{C}_p$*, there exists a budget partition* $\mathbf{b}^* \in \mathbf{B}_p$ *such that* $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b}^*)} \Subset_s \mathcal{AS}$ *for all value partitions* $\mathbf{u} \in \mathbf{U}_p$*;*

(2) for each budget partition $\mathbf{b} \in \mathbf{B}_p$*, there exists a cost partition* $\mathbf{c}^* \in \mathbf{C}_p$ *such that* $\mathcal{M}^{(\mathbf{c}^*,\mathbf{u},\mathbf{b})} \Subset_s \mathcal{AS}$ *for all value partitions* $\mathbf{u} \in \mathbf{U}_p$*.*

Proof: Let $\pi = \langle (s_0, o_1, s_1), (s_1, o_2, s_2), \ldots, (s_{n-1}, o_n, s_n)\rangle$ be an optimal s_0-plan for M_Π, and, for $i \in [k]$, let $\pi_i = \langle (\alpha_i(s_0), o_1, \alpha_i(s_1)), \ldots, (\alpha_i(s_{n-1}), o_n, \alpha_i(o_n))\rangle$ be the mapping of π to G_i. Since $\mathcal{AS}$ is homomorphic, the paths $\pi_1, \ldots, \pi_k$ are well-defined.

(1) Given a cost partition $\mathbf{c} \in \mathbf{C}_p$, let budget profile $\mathbf{b}^* \in \mathbf{B}$ be defined as $\mathbf{b}^*[i] = \sum_{j\in[n]} \mathbf{c}[i](o_j)$, for $i \in [k]$. First, note that $\mathbf{b}^* \in \mathbf{B}_p$ since

$$\sum_{i\in[k]} \mathbf{b}^*[i] = \sum_{i\in[k]} \sum_{j\in[n]} \mathbf{c}[i](o_j) \overset{(\dagger)}{\leq} \sum_{j\in[n]} c(o_j) \overset{(\ddagger)}{\leq} b,$$

where (†) is by $\mathbf{c}$ being a cost partition, and (‡) is by π being an s_0-plan for M_Π. Second, for any $\mathbf{u} \in \mathbf{U}$, by the construction of $\mathbf{b}^*$, π_i is an $\alpha_i(s_0)$-plan for the abstract model $M_i^{(\mathbf{c},\mathbf{u},\mathbf{b}^*)}$. Now, let $\mathbf{u} \in \mathbf{U}_p$, and for $i \in [k]$, let π_i^* be an optimal $\alpha_i(s_0)$-plan for $M_i^{(\mathbf{c},\mathbf{u},\mathbf{b}^*)}$. We have

$$\sum_{i\in[k]} Q^{\mathbf{b}^*[i]}(\pi_i^*) \overset{(\dagger)}{\geq} \sum_{i\in[k]} Q^{\mathbf{b}^*[i]}(\pi_i) \overset{(\ddagger)}{\geq} Q^b(\pi), \tag{21}$$

where (†) is by optimality of π_i^*, and (‡) is by $\alpha_i(s_n)$ being the end-state of π_i and $\mathbf{u}$ being a value partition. Therefore, $(\mathbf{c}, \mathbf{u}, \mathbf{b}^*)$ induces an additive abstraction for Π, that is, $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b}^*)} \Subset_{\mathcal{AS}} M_\Pi$.

(2) Given a budget partition $\mathbf{b}$, let cost profile $\mathbf{c}^* \in \mathbf{C}$ be defined as $\mathbf{c}^*[i](o) = c(o) \cdot \frac{\mathbf{b}[i]}{b}$, for all operators $o \in O$, and all $i \in [k]$. First, we have $\mathbf{c}^* \in \mathbf{C}_p$ since $\mathbf{b} \in \mathbf{B}_p$ implies $\frac{1}{b}\sum_{i\in[k]} \mathbf{b}[i] \in [0,1]$. Second, for any $\mathbf{u} \in \mathbf{U}$, by the construction of $\mathbf{c}^*$, π_i is an $\alpha_i(s_0)$-plan for $M_i^{(\mathbf{c}^*,\mathbf{u},\mathbf{b})}$. Following now exactly the same line of reasoning as for Eq. 21 above accomplishes the proof that $\mathcal{M}^{(\mathbf{c}^*,\mathbf{u},\mathbf{b})} \Subset_{\mathcal{AS}} M_\Pi$ for any $\mathbf{u} \in \mathbf{U}_p$.

$\square$

Lemma 6 *The algorithm in Figure 9a computes* $\kappa(\mathbf{u})$*.*

Proof: Due to the boundness and non-emptiness of the polytope induced by $\mathcal{L}_1^m$, the termination of the algorithm is straightforward. Thus, given a strong 0-binary partition $\mathbf{u}$, the only question is whether the value with which the algorithm terminates is $\kappa(\mathbf{u})$. First, let us show that:

(†) For $m \in [k]$, if $\mathbf{x}$ is a solution of $\mathcal{L}_1^m$, then $\mathbf{x}[\xi] \le b$ if and only if, for *each* cost partition $\mathbf{c} \in \mathbf{C}_p$, there exists a budget partition $\mathbf{b} \in \mathbf{B}_p$ such that $(\mathbf{c},\mathbf{u},\mathbf{b})$ is an abstraction for M_Π and $h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s_0) \ge m\sigma$.

($\Leftarrow$) Assume to the contrary that, for *each* cost partition $\mathbf{c} \in \mathbf{C}_p$, there exists a budget partition $\mathbf{b} \in \mathbf{B}_p$ with $h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s_0) \ge m\sigma$, and yet $\mathbf{x}[\xi] > b$. Given the values provided by $\mathbf{x}$ to the cost variables $\bigcup_{o\in O}\{\mathbb{c}[i](o)\}$, let $\mathbf{c}$ be the corresponding cost partition, and $\delta_1,\ldots,\delta_k$ be the induced lengths of the shortest paths from $\alpha_1(s_0),\ldots,\alpha_k(s_0)$ to σ-valued states in $G_1,\ldots,G_k$, respectively. By our assumption, let $\mathbf{b}$ be a budget partition such that $h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s_0) \ge m\sigma$. First, by the definition of strong 0-binary value partitions, $h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s_0) \ge m\sigma$ implies that there exists $Z \subseteq k$, $|Z| = m$ such that, for $i \in Z$, $\mathbf{b}[i] \ge \delta_i$. Second, constraint (10c), maximization of ξ, and the fact that the only bound on each $\mathbb{b}[i]$ is by δ_i imply together that, for $i \in Z$, $\mathbf{x}[\mathbb{b}[i]] = \delta_i$. Putting things together, we obtain

$$b \overset{\mathbf{b}\in\mathbf{B}_p}{\ge} \sum_{i\in Z}\mathbf{b}[i] \ge \sum_{i\in Z}\delta_i = \sum_{i\in Z}\mathbf{x}[\mathbb{b}[i]] \overset{(10c)}{\ge} \xi,$$

contradicting our assumption.

($\Rightarrow$) Assume to the contrary that, $\mathbf{x}[\xi] \le b$, and yet there exists a cost partition $\mathbf{c} \in \mathbf{C}_p$ such that, for all budget partitions $\mathbf{b} \in \mathbf{B}_p$ with $(\mathbf{c},\mathbf{u},\mathbf{b}) \in \mathbf{A}_p$, we have $h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s_0) < m\sigma$.

Let the shortest path lengths $\delta_1,\ldots,\delta_k$ be defined as above, but now with respect to the specific cost partition $\mathbf{c}$ from the assumption. Likewise, let $\mathbf{x}_\mathbf{c}$ be a solution to $\mathcal{L}_1^m$ with an extra constraint on the cost variables $\bigcup_{o\in O}\{\mathbb{c}[i](o)\}$ to be assigned to $\mathbf{c}$. Since the objective in $\mathcal{L}_1^m$ is to maximize the value of ξ, we have

$$\mathbf{x}[\xi] \ge \mathbf{x}_\mathbf{c}[\xi]. \tag{22}$$

Now, let

$$Z = \operatorname*{argmax}_{Z'\subseteq[k],|Z'|=m} \sum_{i\in Z'}\delta_i.$$

Together, constraint (10c), maximization of ξ, and the fact that the only bound on each $\mathbb{b}[i]$ is by δ_i (via the cost variables) imply that

$$\mathbf{x_c}[\xi] = \sum_{i \in Z} \mathbf{x_c}[\mathbb{b}[i]] = \sum_{i \in Z} \delta_i. \tag{23}$$

In turn, together with $\mathbf{x}[\xi] \leq b$ and Eq. 22, Eq. 23 implies that

$$\mathbf{b}[i] = \begin{cases} \mathbf{x_c}[\mathbb{b}[i]], & i \in Z \\ 0, & \text{otherwise} \end{cases}$$

is a budget partition with $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in \mathbf{A}_p$, and $h_{\mathcal{M}(\mathbf{c},\mathbf{u},\mathbf{b})}(s_0) \geq m\sigma$, contradicting our assumption.

Having proved the sub-claim (†), which basically captures the semantics of $\mathcal{L}_1^m$, suppose that the algorithm terminates within the loop, and returns $m\sigma$ for some $m > 0$. By the construction of the algorithm, if $\mathbf{x}$ is a solution of $\mathcal{L}_1^m$, then $\mathbf{x}[\xi] \leq b$. By (†), for each cost partition $\mathbf{c} \in \mathbf{C}_p$, there exists $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in \mathbf{A}_p$ such that $h_{(\mathbf{c},\mathbf{u},\mathbf{b})}(s) \geq m\sigma$. If $m = k$, then trivially $\kappa_s(\mathbf{u}) = m\sigma$. Otherwise, if $m < k$, we know that the algorithm did not terminate at the previous iteration corresponding to $m+1$. Again, (†) then implies that there exists a cost partition $\mathbf{c} \in \mathbf{C}_p$ for which no $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in \mathbf{A}_p$ will induce $h_{(\mathbf{c},\mathbf{u},\mathbf{b})}(s) \geq (m+1)\sigma$. Hence, by the definition of $\kappa_s(\mathbf{u})$, $\kappa_s(\mathbf{u}) < (m+1)\sigma$, and in turn, since $\mathbf{u}$ is a strong 0-binary value partition, we have $\kappa_s(\mathbf{u}) = m\sigma$. Finally, if the algorithm terminates after the loop and returns 0, then precisely the same argument on the basis of (†) implies $\kappa_s(\mathbf{u}) = 0$. □

Lemma 9 *For any $0 < \epsilon < \min_{i \in [k]} \sigma_i$, the algorithm in Figure 10a computes κ_ϵ such that $\kappa_\epsilon - \kappa(\mathbf{u}) \leq \epsilon$.*

Proof: The arguments for the boundness and non-emptiness of the polytope induced by $\mathcal{L}_2^v$ are precisely the same as for the polytope of $\mathcal{L}_1^m$ studied in Lemma 6, and thus the termination of the algorithm is straightforward. In what follows, we prove that the value returned by the algorithm satisfies the claim of the lemma. Let $\mathbf{u}$ be the given 0-binary partition. Similarly to the proof of Lemma 9, first we prove a sub-claim that:

(†) For $v \in \mathbb{R}^{0+}$, if $\mathbf{x}$ is a solution of $\mathcal{L}_2^v$, then $\mathbf{x}[\xi] \leq b$ if and only if, for *each* cost partition $\mathbf{c} \in \mathbf{C}_p$, there exists a budget partition $\mathbf{b} \in \mathbf{B}_p$ such that $(\mathbf{c}, \mathbf{u}, \mathbf{b})$ is an abstraction for M_Π and $h_{\mathcal{M}(\mathbf{c},\mathbf{u},\mathbf{b})}(s_0) \geq v$.

The proof of (†) mirrors the proof of the respective sub-claim in Lemma 5, *mutatis mutandis*, and thus it is provided here only for ease of verification.

($\Leftarrow$) Assume to the contrary that, for *each* cost partition $\mathbf{c} \in \mathbf{C}_p$, there exists a budget partition $\mathbf{b} \in \mathbf{B}_p$ with $h_{\mathcal{M}(\mathbf{c},\mathbf{u},\mathbf{b})}(s_0) \geq v$, and yet $\mathbf{x}[\xi] > b$.

Given the values provided by $\mathbf{x}$ to the cost variables $\bigcup_{o \in O} \{\mathbb{c}[i](o)\}$, let $\mathbf{c}$ be the corresponding cost partition, and, for $i \in [k]$, let δ_i be the induced length of the shortest path from $\alpha_i(s_0)$ to the σ_i-valued states in G_i. By our assumption, let $\mathbf{b}$ be a budget partition such that $h_{\mathcal{M}(\mathbf{c},\mathbf{u},\mathbf{b})}(s_0) \geq v$. First, by the definition of 0-binary value partitions, $h_{\mathcal{M}(\mathbf{c},\mathbf{u},\mathbf{b})}(s_0) \geq v$ implies that there exists $Z \subseteq k$, $\sum_{i \in Z} \sigma_i \geq v$ such that, for $i \in Z$, $\mathbf{b}[i] \geq \delta_i$.

Second, constraint (11c), maximization of ξ, and the fact that the only bound on each $\mathbb{b}[i]$ is by δ_i, imply together that, for $i \in Z$, $\mathbf{x}[\mathbb{b}[i]] = \delta_i$. Putting things together, we obtain

$$b \overset{\mathbf{b}\in\mathbf{B}_p}{\geq} \sum_{i\in Z} \mathbf{b}[i] \geq \sum_{i\in Z} \delta_i = \sum_{i\in Z} \mathbf{x}[\mathbb{b}[i]] \overset{(11c)}{\geq} \xi,$$

contradicting our assumption.

($\Rightarrow$) Assume to the contrary that, $\mathbf{x}[\xi] \leq b$, and yet there exists a cost partition $\mathbf{c} \in \mathbf{C}_p$ such that, for all budget partitions $\mathbf{b} \in \mathbf{B}_p$ with $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in \mathbf{A}_p$, we have $h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s_0) < v$.

Let the shortest path lengths $\delta_1, \ldots, \delta_k$ be defined as above, but now with respect to the specific cost partition $\mathbf{c}$ from the assumption. Likewise, let $\mathbf{x_c}$ be a solution to $\mathcal{L}_2^v$ with an extra constraint on the cost variables $\bigcup_{o\in O} \{\mathbb{c}[i](o)\}$ to be assigned to $\mathbf{c}$. Since the objective in $\mathcal{L}_2^v$ is to maximize the value of ξ, we have

$$\mathbf{x}[\xi] \geq \mathbf{x_c}[\xi]. \tag{24}$$

Now, let

$$Z = \underset{\substack{Z'\subseteq[k],\\ \sum_{i\in Z'} \sigma_i \geq v}}{\operatorname{argmax}} \sum_{i\in Z'} \delta_i.$$

Together, constraint (11c), maximization of ξ, and the fact that the only bound on each $\mathbb{b}[i]$ is by δ_i (via the cost variables) imply that

$$\mathbf{x_c}[\xi] = \sum_{i\in Z} \mathbf{x_c}[\mathbb{b}[i]] = \sum_{i\in Z} \delta_i. \tag{25}$$

In turn, together with $\mathbf{x}[\xi] \leq b$ and Eq. 24, Eq. 25 implies that

$$\mathbf{b}[i] = \begin{cases} \mathbf{x_c}[\mathbb{b}[i]], & i \in Z \\ 0, & \text{otherwise} \end{cases},$$

is a budget partition with $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in \mathbf{A}_p$, and $h_{\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s_0) \geq v$, contradicting our assumption.

This finalizes the proof of the sub-claim (†). Now, consider the interval end-points α and β at the termination of the while-loop. If $\beta = \sum_{i\in[k]} \sigma_i$, then, trivially, $\kappa(\mathbf{u}) \leq \beta$. Otherwise, if $\beta < \sum_{i\in[k]} \sigma_i$, then, by the construction of the algorithm, at some iteration of the while loop, a test $\mathsf{always\text{-}achievable}(\beta)$ was issued, came back negative, and thus, for the solutions $\mathbf{x}_\beta$ of $\mathcal{L}_2^\beta$, we have $\mathbf{x}_\beta[\xi] > b$. Hence, by (†), $\kappa(\mathbf{u}) < \beta$. Now, if $\alpha \neq 0$, then, by the construction of the algorithm, at some iteration of the while loop, a test $\mathsf{always\text{-}achievable}(\alpha)$ was issued, came back positive, and thus, for the solutions $\mathbf{x}_\alpha$ of $\mathcal{L}_2^\alpha$, we have $\mathbf{x}_\alpha[\xi] \leq b$. Hence, by (†), $\kappa(\mathbf{u}) \geq \alpha$. Putting these properties on α and β together with the while-loop's termination condition $\beta - \alpha \leq \epsilon$ implies $\kappa_\epsilon - \kappa(\mathbf{u}) = \beta - \kappa(\mathbf{u}) \leq \epsilon$. Finally, if $\alpha = 0$, then $\epsilon < \min_{i\in[k]} \sigma_i$ implies $\beta < \min_{i\in[k]} \sigma_i$. In turn, since $\kappa(\mathbf{u})$ corresponds to a sum of values of some states in the k models of $\mathcal{M}^{(\mathbf{c},\mathbf{u},\mathbf{b})}$, $\kappa(\mathbf{u}) \leq \beta$ concluded above implies $\kappa_\epsilon = \kappa(\mathbf{u}) = 0$. $\square$

Theorem 16 *Let $\Pi = \langle V, s_0, u; O, c, b\rangle$ be an OSP task, $\mathcal{L}$ be a set of pairwise disjoint ε-landmarks for Π, lcost be an admissible landmark cost function from $\mathcal{L}$, and $\Pi_{\mathcal{L}}$ be the respective budget reducing compilation of Π. For every π for Π with $Q^b(\pi) > 0$, there is a plan $\pi_{\mathcal{L}}$ for $\Pi_{\mathcal{L}}$ with $Q^{b_{\mathcal{L}}}(\pi_{\mathcal{L}}) = Q^b(\pi)$, and vice versa.*

Proof: Let $\pi_{\mathcal{L}}$ be a plan for $\Pi_{\mathcal{L}}$, and let π be the operator sequence obtained by replacing all operators $\overline{o}$ from $\bigcup_{i=1}^n O_{L_i}$ along $\pi_{\mathcal{L}}$ with the respective operators $o \in O$. By the definition of the action set of $\Pi_{\mathcal{L}}$ in Eq. 15, we have π applicable in s_0, and $s_0[\![\pi]\!] = s_{0\mathcal{L}}[\![\pi_{\mathcal{L}}]\!] \setminus \bigcup_{i=1}^n dom(v_{L_i})$. Thus, $Q^b(\pi) = Q^{b_{\mathcal{L}}}(\pi_{\mathcal{L}})$. Likewise, again by the definition of the action set of $\Pi_{\mathcal{L}}$ in Eq. 15 and the fact that no operator in $O_{\mathcal{L}}$ achieves the control propositions $\{\langle v_{L_1}/1\rangle, \ldots, \langle v_{L_n}/1\rangle\}$, we have $|O_{L_i} \cap \pi_{\mathcal{L}}| \leq 1$. From that, we have

$$c(\pi) \leq c_{\mathcal{L}}(\pi_{\mathcal{L}}) + \sum_{i=1}^{n} lcost(L_i).$$

In turn, $b = b_{\mathcal{L}} + \sum_{i=1}^n lcost(L_i)$ by Eq. 14, and $c_{\mathcal{L}}(\pi_{\mathcal{L}}) \leq b_{\mathcal{L}}$ by the virtue of $\pi_{\mathcal{L}}$ being a plan for $\Pi_{\mathcal{L}}$. Therefore, it holds that $c(\pi) \leq b$, and thus π is a plan for Π.

In the opposite direction, let π be a plan for Π with $Q^b(\pi) > 0$, and let $\pi_{\mathcal{L}}$ be an operator sequence obtained by replacing, for each ε-landmark $L \in \mathcal{L}$, every first occurrence of an operator from L with the respective "cost reduced" operator from O_L. It is easy to verify that $\pi_{\mathcal{L}}$ is applicable in $s_{0\mathcal{L}}$, and that $Q^{b_{\mathcal{L}}}(\pi_{\mathcal{L}}) = Q^b(\pi)$. Likewise, by the definition of ε-landmarks, every $L \in \mathcal{L}$ will have a presence along π. From that, we have

$$c(\pi_{\mathcal{L}}) = c(\pi) - \sum_{i=1}^{n} lcost(L_i) \leq b - \sum_{i=1}^{n} lcost(L_i) = b_{\mathcal{L}},$$

where the first equality is by pairwise disjointness of $\{L_1, \ldots, L_n\}$, the inequality is by π being a plan for Π, and the second equality is by Eq. 14. Thus, $\pi_{\mathcal{L}}$ is a plan for $\Pi_{\mathcal{L}}$. $\square$

Theorem 17 *Let $\Pi = \langle V, s_0, u; O, c, b\rangle$ be an OSP task, let $\mathcal{L} = \{L_1, \ldots, L_n\}$ be a set of ε-landmarks for Π, let lcost be an admissible landmark cost function from $\mathcal{L}$, and let $\Pi_{\mathcal{L}^*}$ be the (generalized) budget reducing compilation of Π. For every π for Π with $Q^b(\pi) > 0$, there is a plan $\pi_{\mathcal{L}^*}$ for $\Pi_{\mathcal{L}^*}$ with $Q^{b_{\mathcal{L}^*}}(\pi_{\mathcal{L}^*}) = Q^b(\pi)$, and vice versa.*

Proof: Let $\pi_{\mathcal{L}^*}$ be a plan for $\Pi_{\mathcal{L}^*}$, and let π be the operator sequence obtained by (i) replacing all operators $\overline{o}$ with the respective operators $o \in O$, and (ii) removal of all *get* operators. By Eq. 17, we have π applicable in s_0, and $s_0[\![\pi]\!] = s_{0\mathcal{L}^*}[\![\pi_{\mathcal{L}^*}]\!] \setminus \{\langle v_{L_1}/1\rangle, \ldots, \langle v_{L_n}/1\rangle\}$. Thus, $Q^b(\pi) = Q^{b_{\mathcal{L}^*}}(\pi_{\mathcal{L}^*})$. Now, for each ε-landmark $L \in \mathcal{L}$, let $\xi(L)$ be the number of instances of the cost reduced counterparts $\overline{o}$ of the operators from L along $\pi_{\mathcal{L}^*}$. By Eqs. 17 and 18, for each $L \in \mathcal{L}$, $\pi_{\mathcal{L}^*}$ must contain at least $\xi(L) - 1$ instances of operator $get(L)$. From that, we have

$$\begin{aligned}
c(\pi) &\leq c_{\mathcal{L}^*}(\pi_{\mathcal{L}^*}) + \sum_{\overline{o} \in \pi_{\mathcal{L}^*}} \sum_{L \in \mathcal{L}(o)} lcost(L) - \sum_{L \in \mathcal{L}} (\xi(L) - 1) \cdot lcost(L) \\
&= c_{\mathcal{L}^*}(\pi_{\mathcal{L}^*}) + \sum_{L \in \mathcal{L}} \xi(L) \cdot lcost(L) - \sum_{L \in \mathcal{L}} (\xi(L) - 1) \cdot lcost(L) \\
&= c_{\mathcal{L}^*}(\pi_{\mathcal{L}^*}) + \sum_{L \in \mathcal{L}} lcost(L) \\
&\leq b_{\mathcal{L}^*} + \sum_{L \in \mathcal{L}} lcost(L) \\
&= b,
\end{aligned}$$

and thus π is a plan for Π.

In the opposite direction, let $\pi = \langle o_1, \ldots, o_m \rangle$ be a plan for Π with $Q^b(\pi) > 0$. By the definition of ε-landmarks, every landmark $L_i \in \mathcal{L}$ will have a presence along π. Let $o_{f(i)}$, $f(i) \in [n]$, be the first occurrence of an operator from L_i along π, that is, $f(i) = \operatorname{argmin}_{j \in [m]}\{o_j \in L_i\}$, and let $\rho = \left\langle o_{(1)}, \ldots, o_{(k)} \right\rangle$, $k \leq n$, be the operator sequence obtained by ordering the operators $\bigcup_{i \in [n]} \{o_{f(i)}\}$ consistently with π. Note that, since ε-landmarks in $\mathcal{L}$ are not necessarily disjoint, we may have $f(i) = f(j)$ for some $1 \leq i \neq j \leq n$, and thus k can be strictly smaller than n.

Given the above, let $\pi_{\mathcal{L}^*}$ be an operator sequence obtained from π based on ρ by

(1) replacing each $o_{(i)}$ along π with $\overline{o}_{(i)}$, and

(2) inserting right before each $\overline{o}_{(i)}$ an arbitrary ordered sequence of actions

$$\bigcup_{j=1}^{i-1} \{get(L) \mid L \in \mathcal{L}, \{o_{(j)}, o_{(i)}\} \subseteq L\}. \tag{26}$$

Note the set union semantics of Eq. 26: even if multiple operators from $\{o_{(1)}, \ldots, o_{(i-1)}\}$ appear in some landmark L together with $o_{(i)}$, only one instance of the operator $get(L)$ is inserted in step (2) before $\overline{o}_{(i)}$.

It is not hard to verify that $\pi_{\mathcal{L}^*}$ is applicable in $s_{0\mathcal{L}^*}$, and that $Q^{b_{\mathcal{L}^*}}(\pi_{\mathcal{L}^*}) = Q^b(\pi)$. Now, step (1) of expanding π to $\pi_{\mathcal{L}^*}$ reduces the cost of the operator sequence by

$$\sum_{i=1}^{k} \sum_{L \in \mathcal{L}(o_{(i)})} lcost(L) = \sum_{L \in \mathcal{L}} \mu(L) \cdot lcost(L),$$

where $\mu(L)$ is the number of all occurrences of the operators from L in ρ. In turn, step (2) of expanding π to $\pi_{\mathcal{L}^*}$ increases the cost of the operator sequence by $\sum_{L \in \mathcal{L}} (\mu(L) - 1) \cdot lcost(L)$. This is because, by Eq. 26, among all $\mu(L)$ operators $\overline{o}_{(i)}$ along $\pi_{\mathcal{L}^*}$ such that $o_{(i)} \in L$, *all but the first* are preceded by the dedicated instances of the operator $get(L)$. Thus,

$$c_{\mathcal{L}^*}(\pi_{\mathcal{L}^*}) = c(\pi) - \sum_{L \in \mathcal{L}} lcost(L) \leq b - \sum_{L \in \mathcal{L}} lcost(L) = b_{\mathcal{L}^*},$$

that is, $\pi_{\mathcal{L}^*}$ is a plan for $\Pi_{\mathcal{L}^*}$. $\square$

Lemma 20 *Let Π be an OSP task, $\Pi_{(\varepsilon, S_{ref})}$ be a (ε, S_{ref})-compilation of Π, $\mathcal{L}$ be a set of landmarks for $\Pi_{(\varepsilon, S_{ref})}$, lcost be an admissible landmark cost function for $\mathcal{L}$, and $\Pi_{\mathcal{L}^*}$ be the respective budget reducing compilation of $(\mathcal{L}, lcost)$ into Π. Let π_1 and π_2 be a pair of plans for $\Pi_{\mathcal{L}^*}$ with end-states s_1 and s_2, respectively, such that $s_1^{\downarrow V} = s_2^{\downarrow V}$ and*

$$c_{\mathcal{L}^*}(\pi_1) + \sum_{L:\langle v_L/0\rangle \in s_1} lcost(L) \;\geq\; c_{\mathcal{L}^*}(\pi_2) + \sum_{L:\langle v_L/0\rangle \in s_2} lcost(L). \tag{20}$$

Then, for any plan π_1' that extends π_1, there exists a plan π_2' that extends π_2 such that $Q^{b_{\mathcal{L}^*}}(\pi_2') = Q^{b_{\mathcal{L}^*}}(\pi_1')$.

Proof: Under the notation in the claim, the proof is by a constructive mapping of the plan π_1' to the corresponding plan π_2'.

First, we derive from π_1' a plan ρ_1' for Π by (i) removing the $finish$ operator and all the $get(\cdot)$ operators, and (ii) replacing all instances of each discounted operator $\overline{o}$ with instances of the respective original operator o. This results in a plan $\rho_1' := \rho_1 \cdot \rho_{1e}$ for Π with $s_0[\![\rho_1]\!] = s_1^{\downarrow V}$ and $c(\rho_1) = c_{\mathcal{L}^*}(\pi_1) + \sum_{L:\langle v_L/0\rangle \in s_1} lcost(L)$. To see the latter, for each operator $\omega \in O_{\mathcal{L}^*}$, let $\kappa(\omega) \geq 0$ denote the number of instances of ω along π_1. Given that, we have

$$\begin{aligned}
c(\rho_1) &= c_{\mathcal{L}^*}(\pi_1) - \sum_{L\in\mathcal{L}} \kappa(get(L)) lcost(L) + \sum_{L\in\mathcal{L}} lcost(L) \sum_{o:L\in\mathcal{L}(o)} \kappa(\overline{o}) \\
&= c_{\mathcal{L}^*}(\pi_1) + \sum_{L\in\mathcal{L}} lcost(L) \left[\left(\sum_{o:L\in\mathcal{L}(o)} \kappa(\overline{o}) \right) - \kappa(get(L)) \right] \\
&= c_{\mathcal{L}^*}(\pi_1) + \sum_{L\in\mathcal{L}} lcost(L) \cdot \mathbb{1}_{s_1}(\langle v_L/0\rangle) \\
&= c_{\mathcal{L}^*}(\pi_1) + \sum_{L:\langle v_L/0\rangle \in s_1} lcost(L),
\end{aligned} \tag{27}$$

where the second and fourth equalities are just formula manipulations, the first equality is direct from the construction of ρ_1, and the third equality is by the definition of the budget reducing compilation, and specifically, by Eqs. 17 and 18.

Similarly to the construction of ρ_1 from π_1, we can construct ρ_2 from π_2, with and $s_0[\![\rho_2]\!] = s_2^{\downarrow V}$ and $c(\rho_2) = c_{\mathcal{L}^*}(\pi_2) + \sum_{L:\langle v_L/0\rangle \in s_2} lcost(L)$. Thus, by Eq. 20, $c(\rho_1) \geq c(\rho_2)$, and also, by the setting of the lemma, $s_0[\![\rho_1]\!] = s_0[\![\rho_2]\!]$. Hence, $\rho_2' = \rho_2 \cdot \rho_{1e}$ is also a plan for Π, and $Q^b(\rho_1') = Q^b(\rho_2')$.

As the last step, we now construct from ρ_2' a plan π_2' for $\Pi_{\mathcal{L}^*}$ as in the claim. First, by the properties of π_2 in the claim, the plan ρ_2 for Π achieves all the landmarks $\mathcal{L}_{\not\in s_2} = \{L \mid \langle v_L/0\rangle \in s_2\}$. Second, by the definition of the landmark set $\mathcal{L}$, ρ_{1e} must satisfy the rest of the landmarks, that is, $\mathcal{L}_{\in s_2} = \{L \mid \langle v_L/1\rangle \in s_2\}$. Let us denote the operator instances along ρ_{1e} as $\langle o_1, \dots, o_k\rangle$, $k = |\rho_{1e}|$, and let $\{\mathcal{L}_1, \dots, \mathcal{L}_k\}$ be a partition of $\mathcal{L}_{\in s_2}$ with $\mathcal{L}_i \subseteq \mathcal{L}_{\in s_2}$ being the subset of all landmarks from $\mathcal{L}_{\in s_2}$ for which o_i is their first achiever along ρ_{1e}.

Given that, consider an operator sequence $\pi_{2e} := \pi^{(k)}$, recursively defined via $\pi^{(0)} = \epsilon$, and, if $\mathcal{L}_i = \emptyset$, then $\pi^{(i)} = \pi^{(i-1)} \cdot \langle o_i\rangle$, else $\pi^{(i)} = \pi^{(i-1)} \cdot \gamma \cdot \langle \overline{o_i}\rangle$, where γ is some (arbitrary) sequencing of operators

$$\{get(L) \mid L \in \mathcal{L}_i \wedge \langle v_L/0\rangle \in s_0[\![\pi_2]\!][\![\pi^{(i-1)}]\!]\}.$$

Finally, we set $\pi'_2 := \pi_2 \cdot \pi_{2e}$.

By Eqs. 17 and 18 in the definition of the budget reducing compilation, it is easy to verify that the above construction of π_{2e} ensures $c_{\mathcal{L}^*}(\pi_{2e}) = c(\rho_{1e}) - \sum_{\langle v_L/1\rangle \in s_2} lcost(L)$ and $Q^{b_{\mathcal{L}^*}}(\pi_{2e}) = Q^b(\rho'_2)$. In turn, by the properties of π_2, this implies that $Q^{b_{\mathcal{L}^*}}(\pi'_2) = Q^{b_{\mathcal{L}^*}}(\pi'_1)$ and $c_{\mathcal{L}^*}(\pi'_2) = c_{\mathcal{L}^*}(\pi_2) + c_{\mathcal{L}^*}(\pi_{2e})$.

Finally, since

$$c_{\mathcal{L}^*}(\pi_2) = c(\rho_2) - \sum_{\langle v_L/0\rangle \in s_2} lcost(L)$$

and

$$c_{\mathcal{L}^*}(\pi_{2e}) = c(\rho_{1e}) - \sum_{\langle v_L/1\rangle \in s_2} lcost(L),$$

we have

$$c_{\mathcal{L}^*}(\pi'_2) = c(\rho_2) + c(\rho_{1e}) - \sum_{L \in \mathcal{L}} lcost(L).$$

Thus, since $c(\rho_1) \geq c(\rho_2)$ and $\rho'_1 = \rho_1 \cdot \rho_{1e}$ is a valid plan for Π, we have

$$\begin{aligned} c_{\mathcal{L}^*}(\pi'_2) &\leq c(\rho_1) + c(\rho_{1e}) - \sum_{L \in \mathcal{L}} lcost(L) \\ &\leq c(\rho'_1) - \sum_{L \in \mathcal{L}} lcost(L) \\ &\leq b - \sum_{L \in \mathcal{L}} lcost(L), \end{aligned}$$

finalizing the proof that π'_2 is a plan for $\Pi_{\mathcal{L}^*}$ as in the claim. $\square$

Appendix B. Detailed Evaluation Results

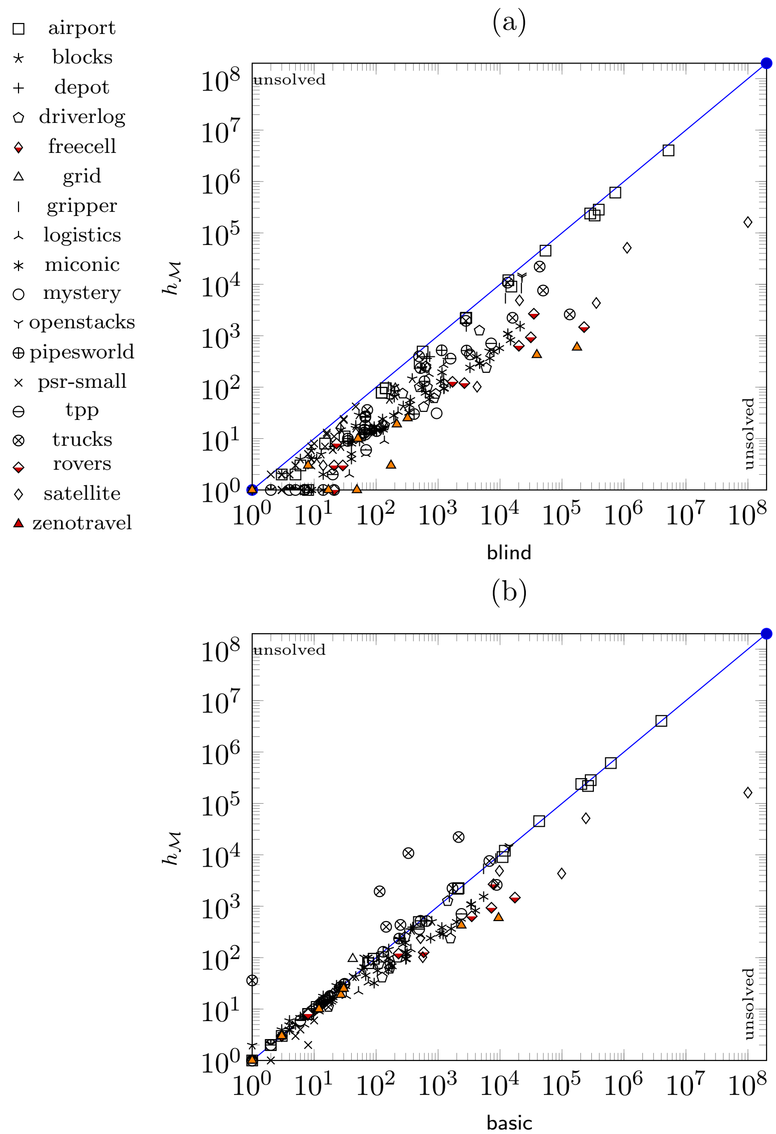


Figure 18: The comparison in Figure 8, p. 117, restricted to the tasks budgeted with 25% of the minimal cost of achieving the entire set of subgoals

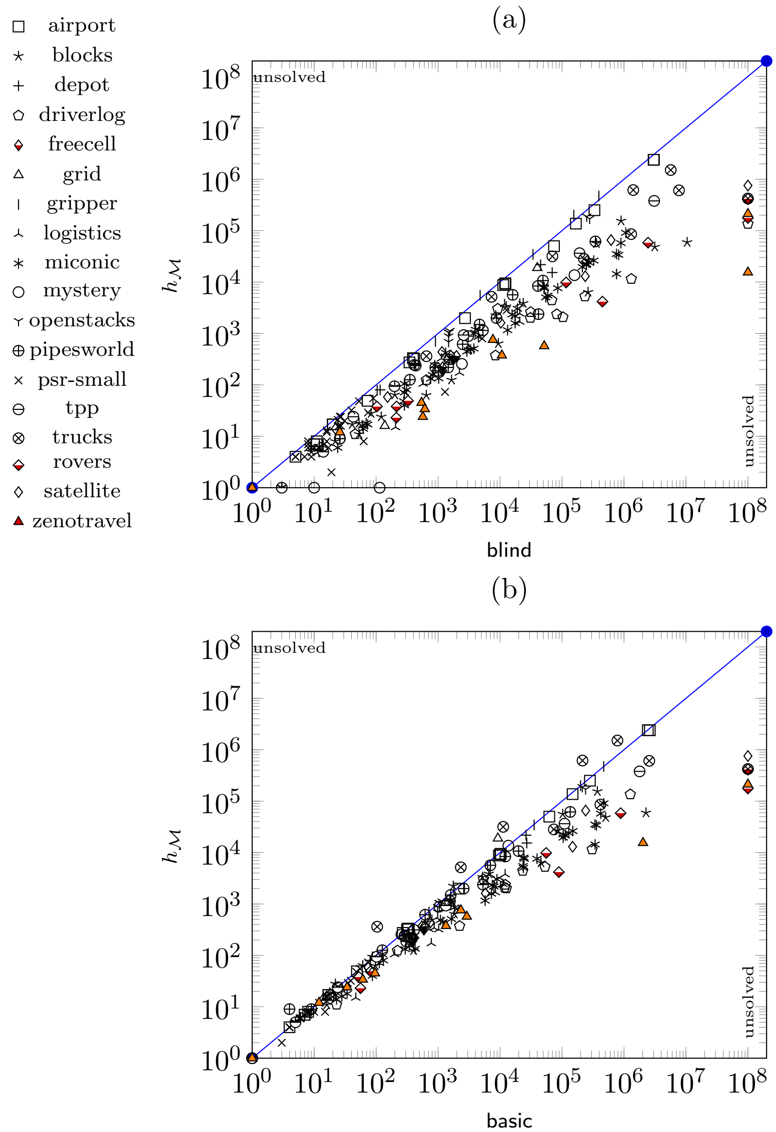


Figure 19: The comparison in Figure 8, p. 117, restricted to the tasks budgeted with 50% of the minimal cost of achieving the entire set of subgoals

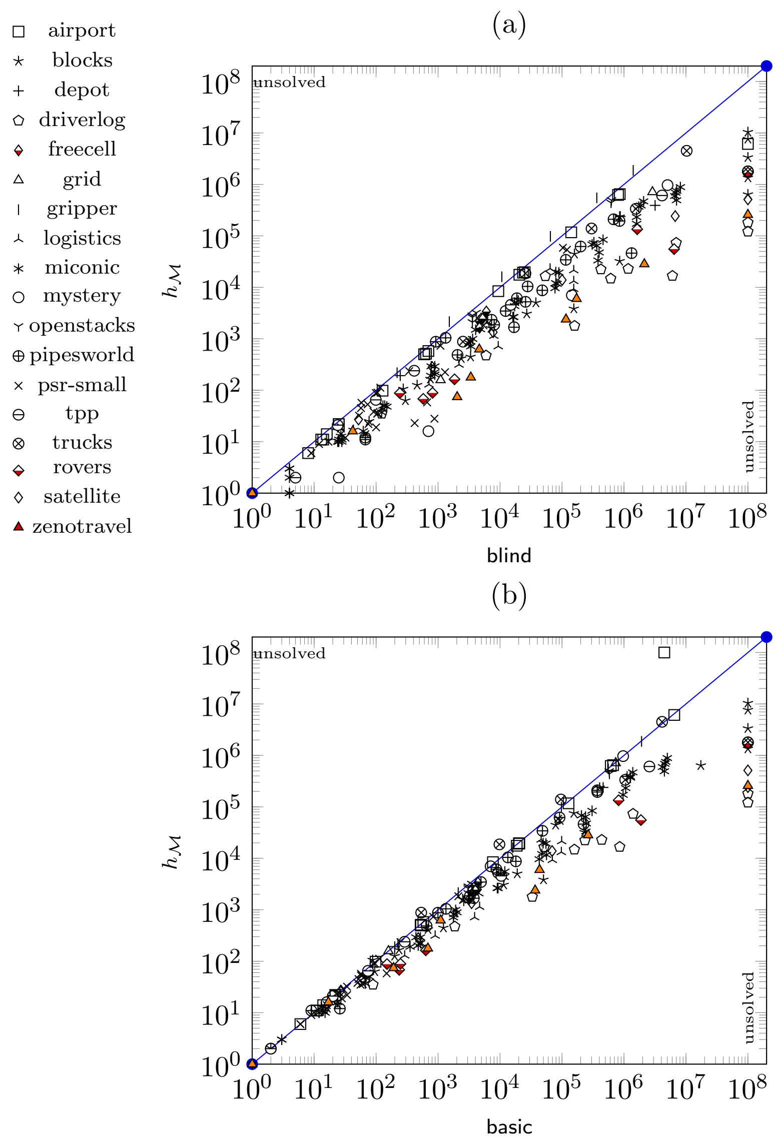


Figure 20: The comparison in Figure 8, p. 117, restricted to the tasks budgeted with 75% of the minimal cost of achieving the entire set of subgoals

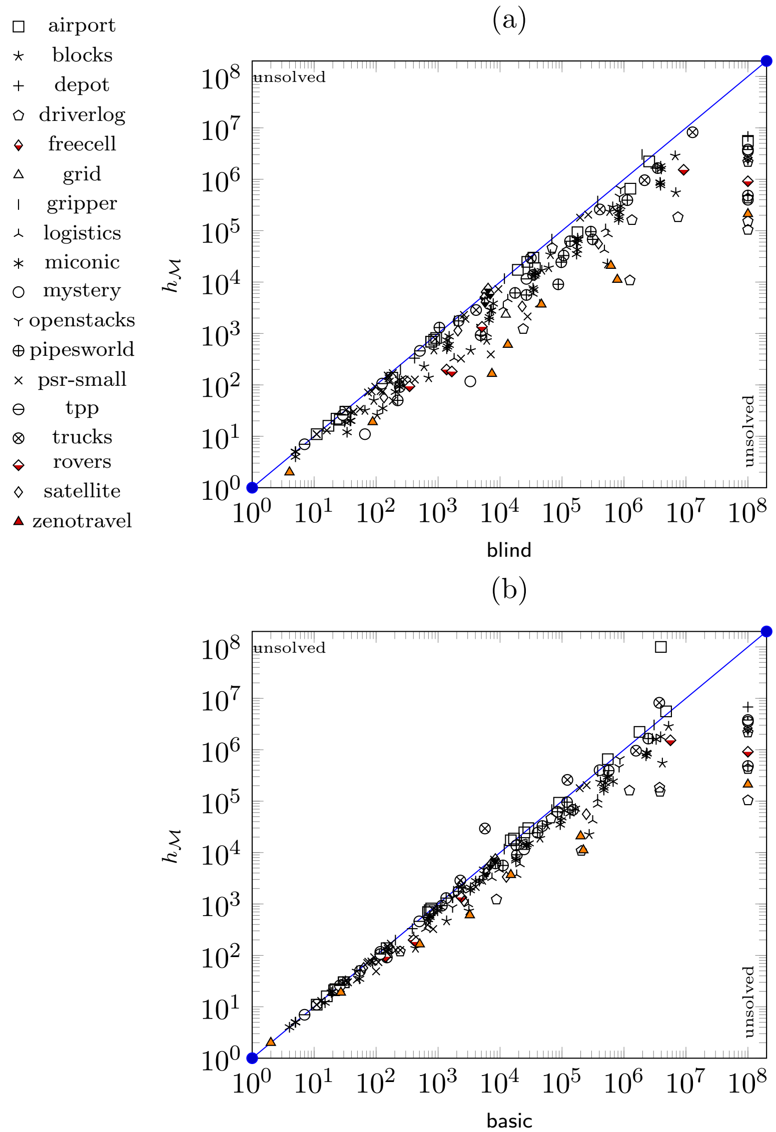


Figure 21: The comparison in Figure 8, p. 117, restricted to the tasks budgeted with 100% of the minimal cost of achieving the entire set of subgoals

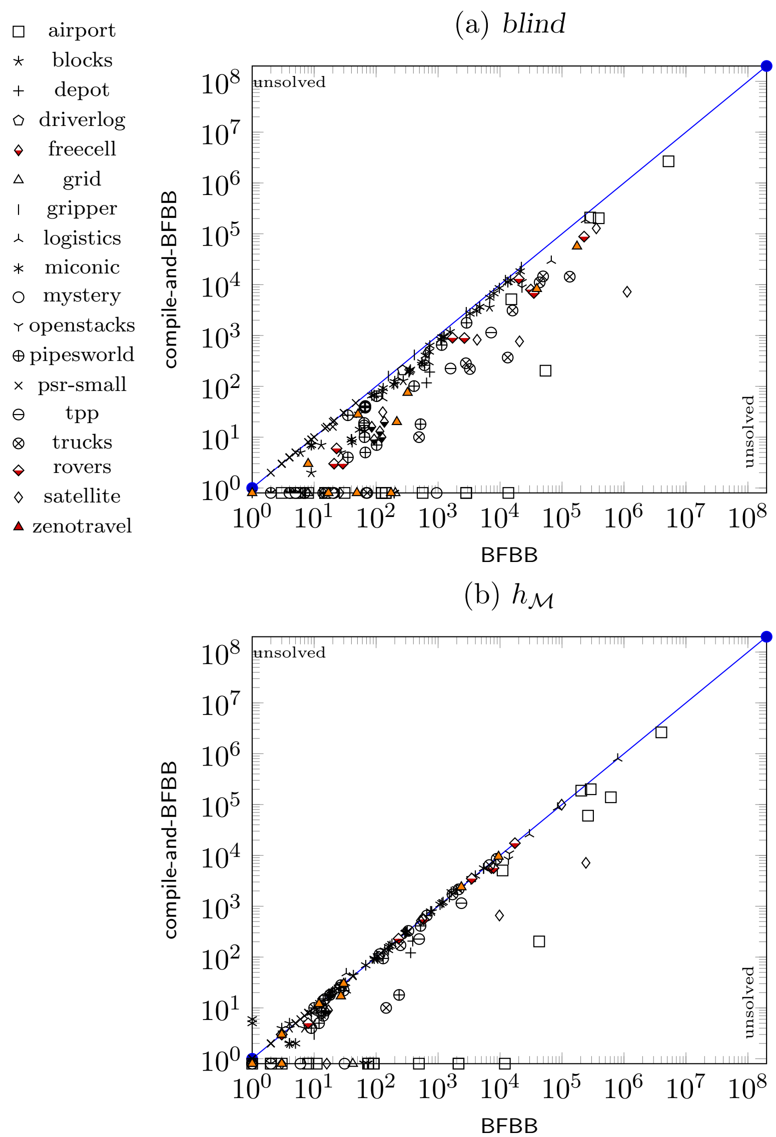


Figure 22: The comparison in Figure 15, p. 140, restricted to the tasks budgeted with 25% of the minimal cost of achieving the entire set of subgoals

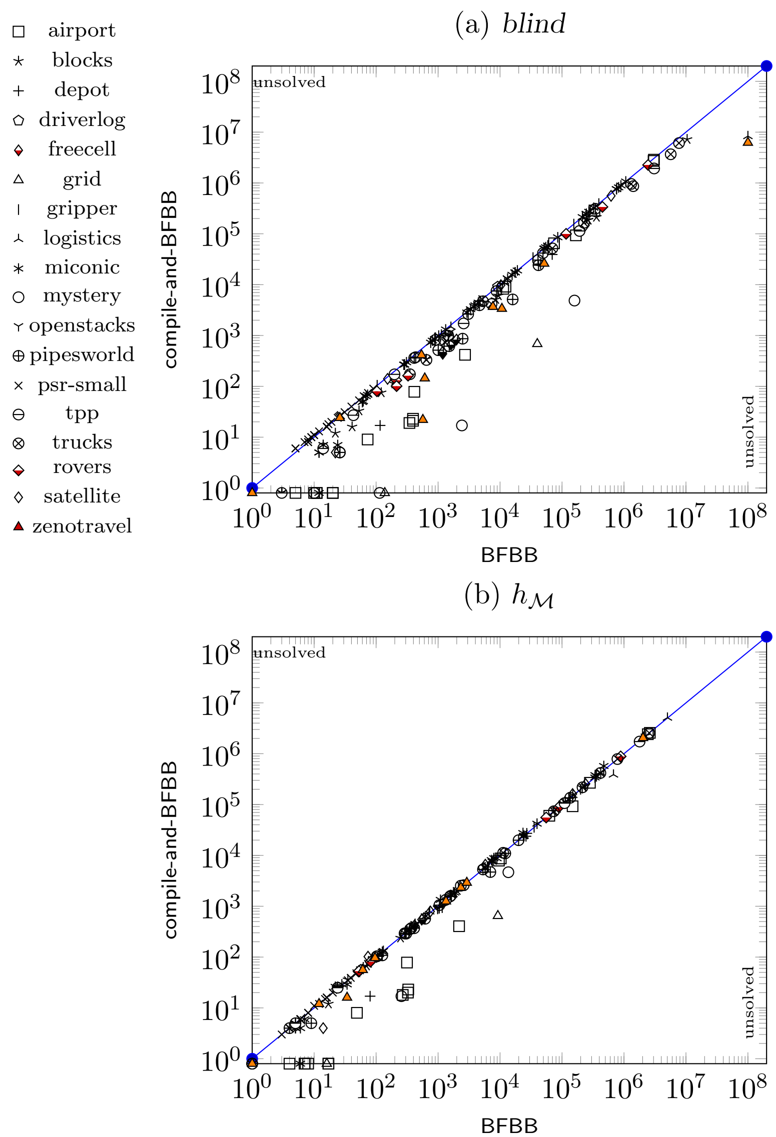


Figure 23: The comparison in Figure 15, p. 140, restricted to the tasks budgeted with 50% of the minimal cost of achieving the entire set of subgoals

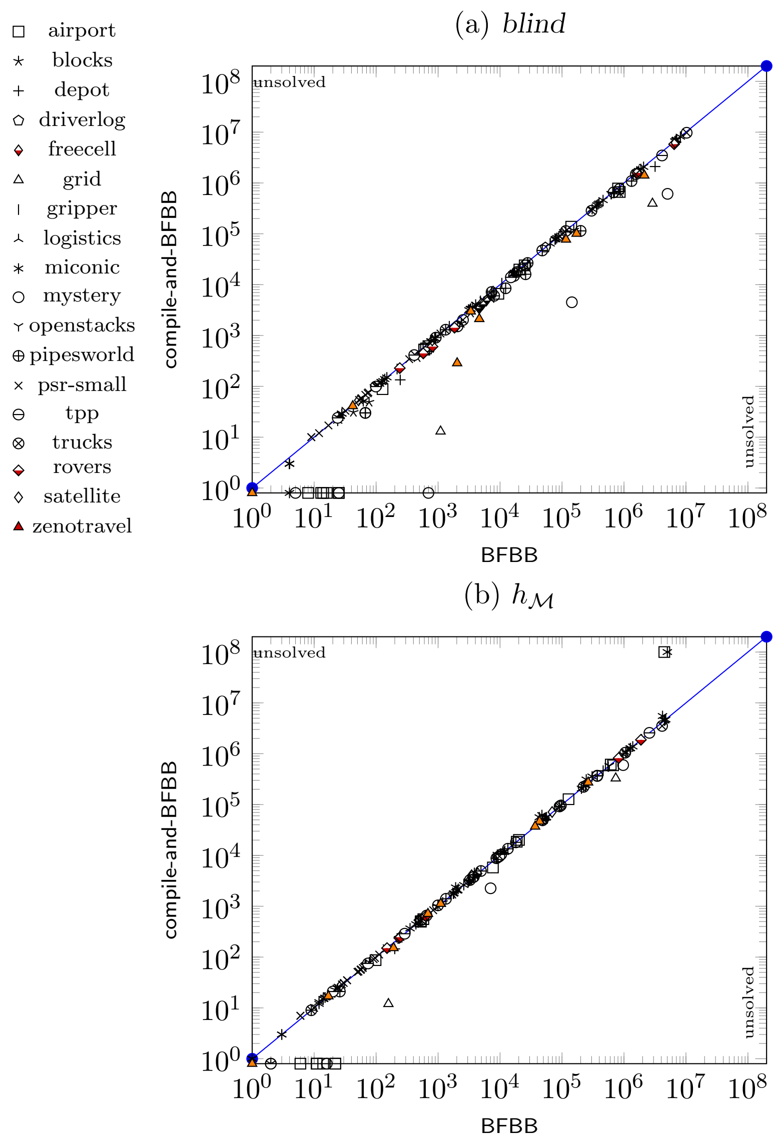


Figure 24: The comparison in Figure 15, p. 140, restricted to the tasks budgeted with 75% of the minimal cost of achieving the entire set of subgoals

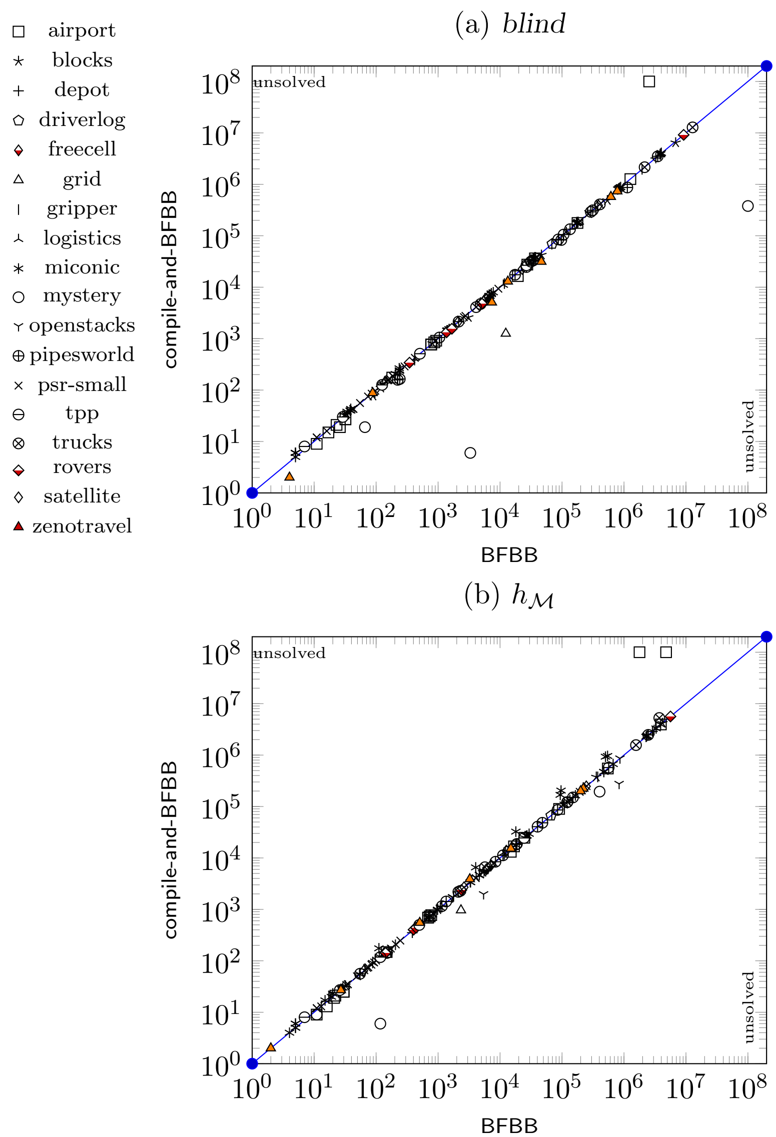


Figure 25: The comparison in Figure 15, p. 140, restricted to the tasks budgeted with 100% of the minimal cost of achieving the entire set of subgoals

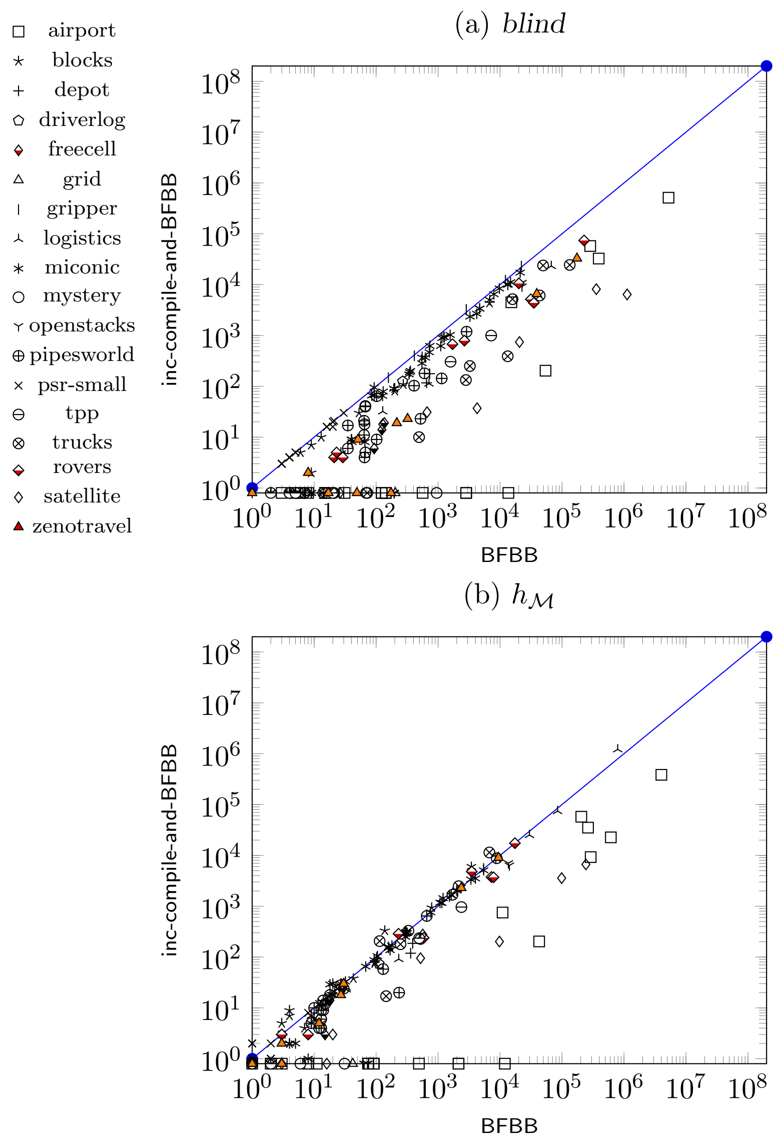


Figure 26: The comparison in Figure 16, p. 142, restricted to the tasks budgeted with 25% of the minimal cost of achieving the entire set of subgoals

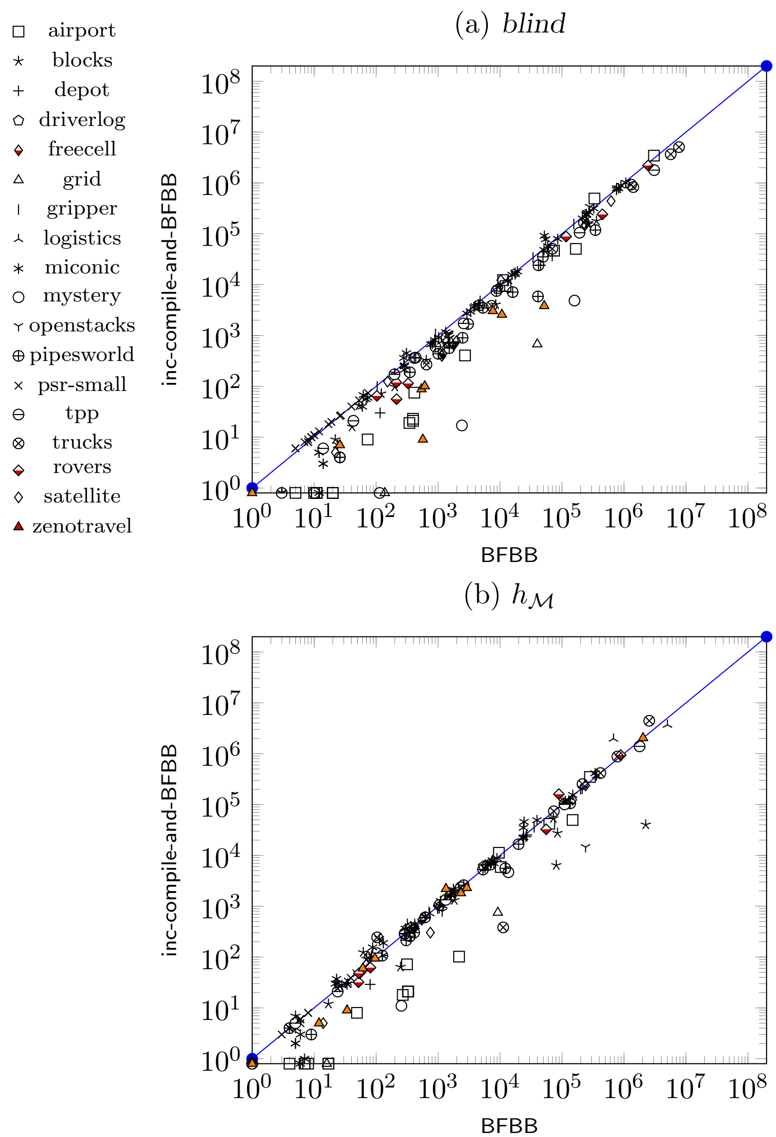


Figure 27: The comparison in Figure 16, p. 142, restricted to the tasks budgeted with 50% of the minimal cost of achieving the entire set of subgoals

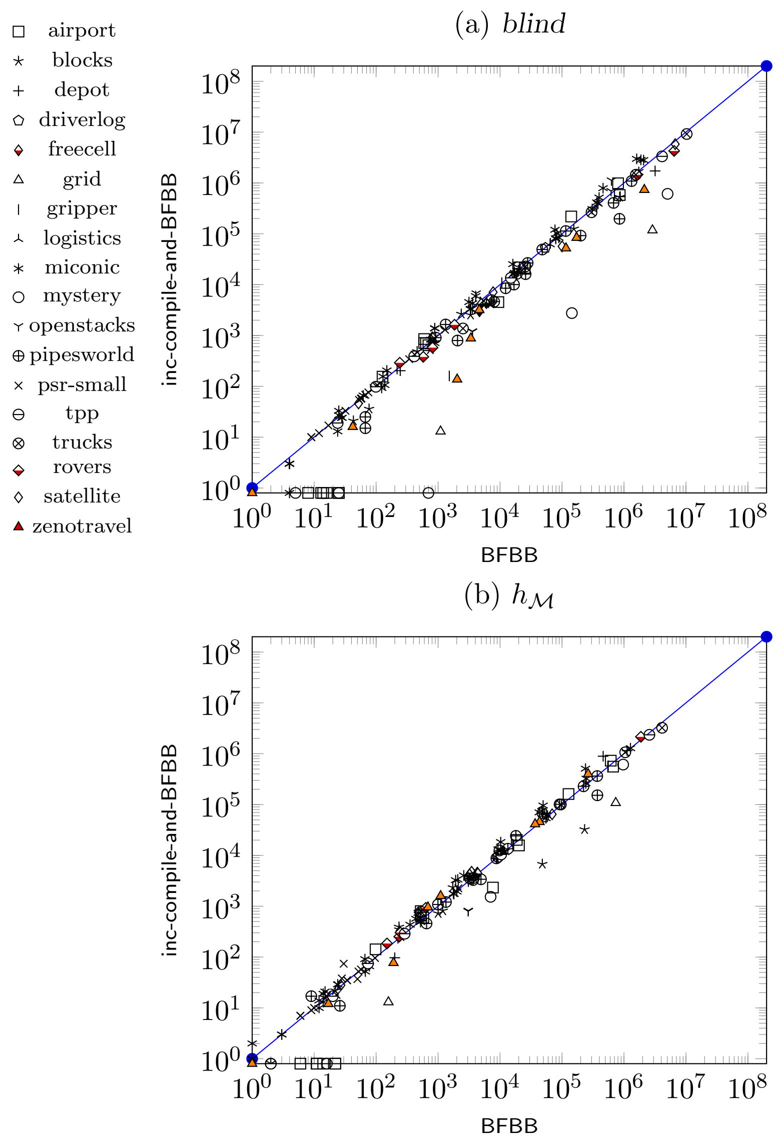


Figure 28: The comparison in Figure 16, p. 142, restricted to the tasks budgeted with 75% of the minimal cost of achieving the entire set of subgoals

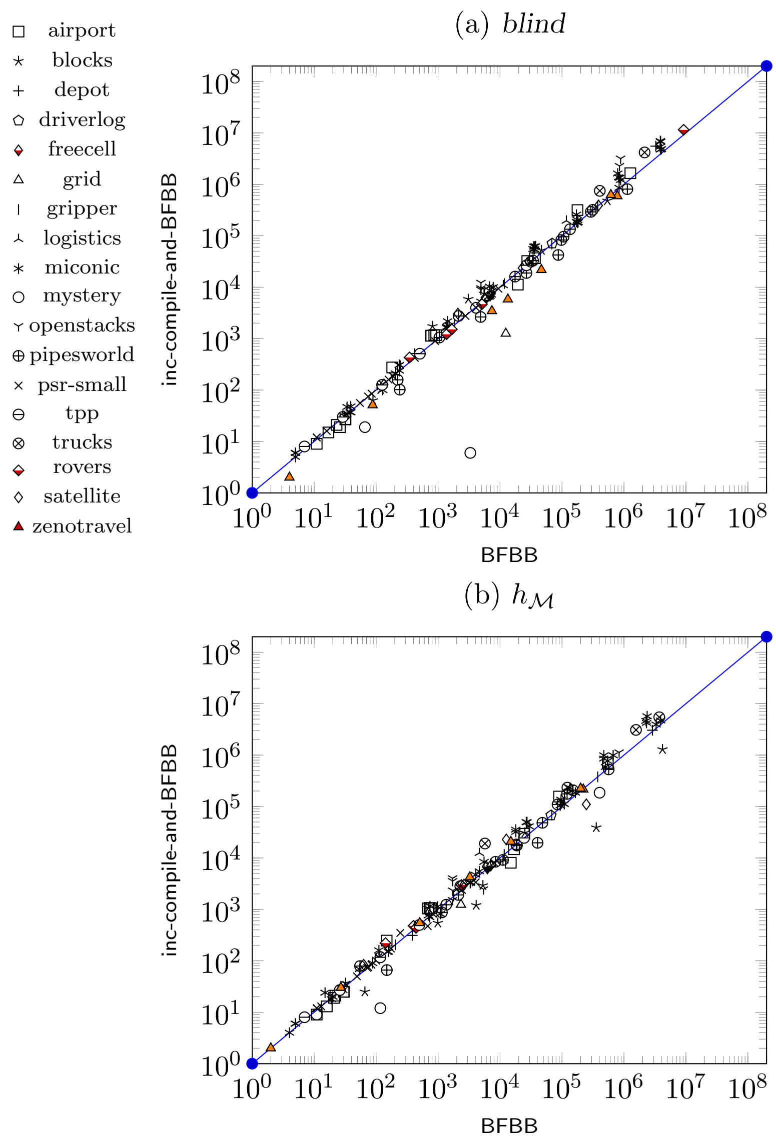


Figure 29: The comparison in Figure 16, p. 142, restricted to the tasks budgeted with 100% of the minimal cost of achieving the entire set of subgoals

References

Bäckström, C., & Klein, I. (1991). Planning in polynomial time: The SAS-PUBS class. *Computational Intelligence*, *7*(3), 181–197.

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

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

Benton, J., Coles, A. J., & Coles, A. I. (2012). Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 2–10.

Benton, J., Do, M., & Kambhampati, S. (2009). Anytime heuristic search for partial satisfaction planning. *Artificial Intelligence*, *173*(5-6), 562–592.

Benton, J., van den Briel, M., & Kambhampati, S. (2007). A hybrid linear programming and relaxed plan heuristic for partial satisfaction planning problems. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 34–41.

Bertsimas, D., & Vempala, S. (2004). Solving convex programs by random walks. *Journal of the ACM*, *51*(4), 540–556.

Bonet, B. (2013). An admissible heuristic for SAS$^+$ planning obtained from the state equation. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2268–2274.

Bonet, B., & Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, *129*(1–2), 5–33.

Bonet, B., & Geffner, H. (2008). Heuristics for planning with penalties and rewards formulated in logic and computed through circuits. *Artificial Intelligence*, *172*(12-13), 1579–1604.

Bonet, B., & Helmert, M. (2010). Strengthening landmark heuristics via hitting sets. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI)*, pp. 329–334.

Brafman, R. I., & Chernyavsky, Y. (2005). Planning with goal preferences and constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pp. 182–191.

Clarke, E., Grumberg, O., & Peled, D. (1999). *Model Checking*. MIT Press.

Coles, A. I., Fox, M., Long, D., & Smith, A. J. (2008). Additive-disjunctive heuristics for optimal planning. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 44–51.

Coles, A. J., Coles, A., Fox, M., & Long, D. (2013). A hybrid LP-RPG heuristic for modelling numeric resource flows in planning. *Journal of Artificial Intelligence Research*, *46*, 343–412.

Coles, A. J., & Coles, A. I. (2011). LPRPG-P: Relaxed plan heuristics for planning with preferences. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 37–45.

Cousot, P., & Cousot, R. (1992). Abstract interpretation frameworks. *Journal of Logic and Computation*, *2*(4), 511–547.

Dantzig, G. B. (1963). *Linear Programming and Extensions*. Princeton University Press.

Dantzig, T. (1930). *Number: The Language of Science*. Macmillan.

Desrochers, M., & Soumis, F. (1988). A generalized permanent labelling algorithm for the shortest path problem with time windows. *Information Systems and Operations Research*, *26*, 191–212.

Do, M. B., Benton, J., van den Briel, M., & Kambhampati, S. (2007). Planning with goal utility dependencies. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1872–1878.

Domshlak, C., Hoffmann, J., & Sabharwal, A. (2009). Friends or foes? On planning as satisfiability and abstract CNF encodings. *Journal of Artificial Intelligence Research*, *36*, 415–469.

Domshlak, C., Katz, M., & Lefler, S. (2012). Landmark-enhanced abstraction heuristics. *Artificial Intelligence*, *189*, 48–68.

Dudzinski, K., & Walukiewicz, S. (1987). Exact methods for the Knapsack problem and its generalizations. *European Journal of Operational Research*, *28*, 3–21.

Dvorak, F., & Barták, R. (2010). Integrating time and resources into planning. In *Proceedings of the 22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 71–78.

Edelkamp, S. (2001). Planning with pattern databases. In *Proceedings of the European Conference on Planning (ECP)*, pp. 84–90.

Edelkamp, S. (2003). Taming numbers and durations in the model checking integrated planning system. *Journal of Artificial Intelligence Research*, *20*, 195–238.

Fikes, R. E., & Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*, 189–208.

Fox, M., & Long, D. (2003). PDDL2.1: An extension to PDDL for expressing temporal planning problems. *Journal of Artificial Intelligence Research*, *20*, 61–124.

Garey, M. R., & Johnson, D. S. (1978). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York.

Geffner, H., & Bonet, B. (2013). *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool.

Gerevini, A., Haslum, P., Long, D., Saetti, A., & Dimopoulos, Y. (2009). Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, *173*(5-6), 619–668.

Gerevini, A., Saetti, A., & Serina, I. (2003). Planning through stochastic local search and temporal action graphs in LPG. *Journal of Artificial Intelligence Research*, *20*, 239–290.

Gerevini, A., Saetti, A., & Serina, I. (2008). An approach to efficient planning with numerical fluents and multi-criteria plan quality. *Artificial Intelligence*, *172*(8-9), 899–944.

Grotschel, M., Lovasz, L., & Schrijver, A. (1981). The ellipsoid method and its consequences theorems in combinatorial optimization. *Combinatorica*, *1*, 169–197.

Handler, G., & Zang, I. (1980). A dual algorithm for the constrained shortest path problem. *Networks*, *10*, 293–310.

Haslum, P. (2013). Heuristics for bounded-cost search. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 312–316.

Haslum, P., Bonet, B., & Geffner, H. (2005). New admissible heuristics for domain-independent planning. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*, pp. 1163–1168.

Haslum, P., Botea, A., Helmert, M., Bonet, B., & Koenig, S. (2007). Domain-independent construction of pattern database heuristics for cost-optimal planning. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI)*, pp. 1007–1012.

Haslum, P., & Geffner, H. (2000). Admissible heuristics for optimal planning. In *Proceedings of the 15th International Conference on Artificial Intelligence Planning Systems (AIPS)*, pp. 140–149.

Haslum, P., & Geffner, H. (2001). Heuristic planning with time and resources. In *Proceedings of the 6th European Conference on Planning (ECP)*, pp. 107–112.

Helmert, M. (2002). Decidability and undecidability results for planning with numerical state variables. In *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, pp. 44–53.

Helmert, M. (2006). The Fast Downward planning system. *Journal of Artificial Intelligence Research*, *26*, 191–246.

Helmert, M., & Domshlak, C. (2009). Landmarks, critical paths and abstractions: What's the difference anyway?. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 162–169.

Helmert, M., Haslum, P., & Hoffmann, J. (2007). Flexible abstraction heuristics for optimal sequential planning. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 200–207.

Helmert, M., Haslum, P., Hoffmann, J., & Nissim, R. (2014). Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the ACM*, *61*(3), 16:1–63.

Hoffmann, J. (2003). The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research*, *20*, 291–341.

Hoffmann, J., Gomes, C. P., Selman, B., & Kautz, H. A. (2007). SAT encodings of state-space reachability problems in numeric domains. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1918–1923.

Hoffmann, J., & Nebel, B. (2001). The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, *14*, 253–302.

Hoffmann, J., Porteous, J., & Sebastia, L. (2004). Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, *22*, 215–278.

Karp, R. (1972). Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pp. 85–103. Plenum Press, New York.

Karpas, E., & Domshlak, C. (2009). Cost-optimal planning with landmarks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-09)*, pp. 1728–1733.

Katz, M., & Domshlak, C. (2010a). Implicit abstraction heuristics. *Journal of Artificial Intelligence Research*, *39*, 51–126.

Katz, M., & Domshlak, C. (2010b). Optimal admissible composition of abstraction heuristics. *Artificial Intelligence*, *174*, 767–798.

Kellerer, H., Pferschy, U., & Pisinger, D. (2004). *Knapsack Problems*. Springer-Verlag Berlin.

Keyder, E., & Geffner, H. (2009). Soft goals can be compiled away. *Journal of Artificial Intelligence Research*, *36*, 547–556.

Koehler, J. (1998). Planning under resource constraints. In *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI)*, pp. 489–493.

Mirkis, V., & Domshlak, C. (2013). Abstractions for oversubscription planning. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 153–161.

Mirkis, V., & Domshlak, C. (2014). Landmarks in oversubscription planning. In *Proceedings of the 23rd European Conference on Artificial Intelligence (ECAI)*, pp. 633–638.

Nakhost, H., Hoffmann, J., & Müller, M. (2012). Resource-constrained planning: A Monte Carlo random walk approach. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 181–189.

Nebel, B. (2000). On the compilability and expressive power of propositional planning formalisms. *Journal of Artificial Intelligence Research*, *12*, 271–315.

Nemirovsky, A., & Yudin, N. (1994). *Interior-Point Polynomial Methods in Convex Programming*. SIAM.

Pearl, J. (1984). *Heuristics - Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

Pommerening, F., & Helmert, M. (2013). Incremental LM-Cut. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 162–170, Rome, Italy.

Porteous, J., Sebastia, L., & Hoffmann, J. (2001). On the extraction, ordering, and usage of landmarks in planning. In *Proceedings of the 6th European Conference on Planning (ECP 01)*, pp. 37–49.

Punnen, A. P. (1992). K-sum linear programming. *The Journal of the Operational Research Society*, *43*(4), 359–363.

Richter, S., Helmert, M., & Westphal, M. (2008). Landmarks revisited. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI-08)*, pp. 975–982.

Russell, S., & Norvig, P. (2009). *Artificial Intelligence: A Modern Approach* (3 edition). Pearson.

Sanchez, R., & Kambhampati, S. (2005). Planning graph heuristics for selecting objectives in over-subscription planning problems. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 192–201.

Smith, D. (2004). Choosing objectives in over-subscription planning. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 393–401.

Thayer, J. T., & Ruml, W. (2011). Bounded suboptimal search: A direct approach using inadmissible estimates. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 674–679.

Thayer, J. T., Stern, R. T., Felner, A., & Ruml, W. (2012). Faster bounded-cost search using inadmissible estimates. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 270–278.

van den Briel, M., Sanchez, R., Do, M. B., & Kambhampati, S. (2004). Effective approaches for partial satisfaction (over-subscription) planning. In *Proceedings of the 19th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 562–569.

van den Briel, M., Benton, J., Kambhampati, S., & Vossen, T. (2007). An LP-based heuristic for optimal planning. In *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming (CP)*, pp. 651–665.

Yang, F., Culberson, J., Holte, R., Zahavi, U., & Felner, A. (2008). A general theory of additive state space abstractions. *Journal of Artificial Intelligence Research*, *32*, 631–662.