Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling

Abstractions for Oversubscription Planning

Vitaly Mirkis and **Carmel Domshlak**
Technion - Israel Institute of Technology
Haifa, Israel
{dcarmel,mirkis}@tx.technion.ac.il

Abstract

In deterministic OSP, the objective 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 this objective, no substantial algorithmic advances have been made beyond the very special settings of net-benefit optimization. Tracing the key sources of progress in classical planning, we identify a severe lack of domain-independent approximations for OSP, and start with investigating the prospects of abstraction approximations for this problem. In particular, 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.

Introduction

In deterministic planning, the basic structure of acting with underconstrained or overconstrained resources is respectively captured by *classical* planning and *oversubscription* planning. In classical planning, all goals must be achieved at as low a total cost of the actions as possible. In oversubscription planning (OSP), an as valuable as possible subset of goals should be achieved within a fixed allowance of the total action cost. While both theory and practice of classical planning have been rapidly advancing, progress in OSP has been mostly in the direction of net-benefit planning. In net-benefit planning, no explicit restriction is put on the plan cost, and the action costs and goal utilities are assumed to be comparable, with the objective being maximizing the difference between the cumulative value of the achieved goals and the cost invested in achieving them. Although there are numerous interesting algorithms for net-benefit planning, it was recently shown to be polynomial-time reducible to classical planning (Keyder and Geffner 2009). As such, it constitutes an extremely special variant of oversubscription.

A closer look shows that the recent progress in classical planning stems, to a large extent, from advances in domain-independent approximations, or heuristics, of the cost needed to achieve all the goals 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. 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 numeric state variables (Fox and Long 2003; Helmert 2002). So far, however, progress in classical planning with numeric state variables has mostly been achieved along delete relaxation heuristics (Hoffmann 2003; Edelkamp 2003), and these heuristics do not preserve information on consumable resources: 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. 2008).

In this work we make first steps towards effective heuristics for OSP, and in particular, towards admissible *abstraction heuristics* for this problem. In classical planning, state-space abstractions are among the most prominent techniques for devising admissible heuristics (Edelkamp 2002; Haslum et al. 2007; Helmert, Haslum, and Hoffmann 2007; Katz and Domshlak 2010a). Departing from the most basic question of what state-space abstractions for OSP actually are (and what they are not), we show that the very notion of abstraction substantially differs in classical and in OSP. We define additive abstractions and abstraction heuristics for OSP, and investigate 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, we expose an interesting interplay between knapsack-style problems, convex optimization, and principles borrowed from explicit abstractions for classical planning. We believe that this interplay opens the road to much further research.

Formalism and Background

In line with the SAS$^+$ formalism for deterministic planning (Bäckström and Klein 1991; Bäckström and Nebel 1995), a ***planning task structure*** is given by a pair $\langle V, A\rangle$, where V is a set of n finite-domain *state variables*, and A is a finite set of *actions*. Each complete assignment to V is called a *state*, and $S = dom(v_1) \times \cdots \times dom(v_n)$ is the *state space* of the structure $\langle V, A\rangle$. Each action a is a pair $\langle \mathsf{pre}(a), \mathsf{eff}(a)\rangle$ of partial assignments to V called *preconditions* and *effects*, respectively. Denoting by $\mathcal{V}(p) \subseteq V$ the subset of variables instantiated by a partial assignment p, action a is applicable in a state s iff $s[v] = \mathsf{pre}(a)[v]$ for

all $v \in \mathcal{V}(\mathsf{pre}(a))$. Applying a changes the value of each $v \in \mathcal{V}(\mathsf{eff}(a))$ to $\mathsf{eff}(a)[v]$. The resulting state is denoted by $s[\![a]\!]$; by $s[\![\langle a_1, \ldots, a_k \rangle]\!]$ we denote the state obtained from sequential application of the (applicable in turn) actions $a_1, \ldots, a_k$ starting at state s.

In classical planning, a planning task $\Pi = \langle V, A; s_0, G, c \rangle$ extends its structure with an initial state $s_0 \in S$, a goal specification G, typically modeled as a partial assignment to V, and an action cost function $c : A \to \mathbb{R}^{0+}$. An action sequence ρ is called an s-plan if it is applicable in s, and $G \subseteq s[\![\rho]\!]$. An s-plan is optimal if the sum of its action costs is minimal among all s-plans. The objective in classical planning is to find an s_0-plan of as low cost as possible, with optimal classical planning being devoted to searching for optimal s_0-plans only.

In contrast, a ***oversubscription planning (OSP) task*** $\Pi = \langle V, A; s_0, c, u, b \rangle$ extends its structure with four components: an *initial state* $s_0 \in S$ and an *action cost function* $c : A \to \mathbb{R}^{0+}$ as above, plus a succinctly represented and efficiently computable *state value* function $u : S \to \mathbb{R}^{0+}$, and a *cost budget* $b \in \mathbb{R}^{0+}$. An action sequence ρ is called an *s-plan* if it is applicable in s, and $\sum_{a \in \rho} c(a) \leq b$; by $\hat{u}(\rho)$ we refer to the value of the end-state of ρ, that is, $\hat{u}(\rho) = u(s[\![\rho]\!])$. While empty action sequence is an s-plan for any state s, the objective in oversubscription planning is to find an s_0-plan that achieves as valuable a state as possible, and ***optimal oversubscription planning*** is devoted to searching for optimal s_0-plans only: An s-plan ρ is *optimal* if $\hat{u}(\rho)$ is maximal among all the s-plans, and if ρ is optimal, then $h^*(s) \stackrel{\mathsf{def}}{=} \hat{u}(\rho)$.

Each planning task Π induces a state-transition model, or transition graph. Following Katz and Domshlak (2010b), we distinguish between the actual node/edge-weighted transition graphs, and their weights-omitted, qualitative skeletons, referred to as transition graph structures. Informally, the latter capture the dynamics of the planning tasks, while the former associate these dynamics with "performance measures" (Russell and Norvig 2009). A ***transition graph structure*** (or ***tg-structure***) is a triplet $\mathcal{T} = \langle S, L, Tr \rangle$ where S is the finite set of states, L is the finite set of labels, and $Tr \subseteq S \times L \times S$ is a set of labeled state transitions. Each tg-structure $\mathcal{T} = \langle S, L, Tr \rangle$ implicitly defines a ***space of performance measures*** that can be associated with it. In the context of OSP, this space constitutes $C \times U \times B$ where C is the set of all functions from labels L to $\mathbb{R}^{0+}$, U is the set of all functions from states S to $\mathbb{R}^{0+}$, and $B = \mathbb{R}^{0+}$. A ***transition graph*** (or ***t-graph***) $\Phi = \langle \mathcal{T}, c, u, b \rangle$ associates a tg-structure $\mathcal{T}$ with a specific performance measure $(c, u, b) \in C \times U \times B$. A path from state s along the transitions of $\mathcal{T}$ is an s-plan for Φ if $\sum_{(s,l,s') \in \pi} c(l) \leq b$.

The tg-structure $\mathcal{T}(\Pi)$ induced by a planning task $\Pi = \langle V, A; s_0, c, u, b \rangle$ is induced by the structure $\langle V, A \rangle$ of the latter: the states and labels of $\mathcal{T}(\Pi)$ are states $S = dom(V)$ and actions A of Π, respectively, and $(s, a, s[\![a]\!]) \in Tr$ iff action a is applicable in state s. The t-graph induced by a planning task $\Pi = \langle V, A; s_0, c, u, b \rangle$ is $\Phi(\Pi) = \langle \mathcal{T}(\Pi), c, u, b \rangle$. Since there is an obvious correspondence between the s-plans for Π and the s-plans for $\Phi(\Pi)$, searching in $\Phi(\Pi)$ corresponds to planning for Π via state-space search, and heuristic-search such procedures employ heuristic functions to estimate the relative attractiveness of various parts of the t-graph $\Phi(\Pi)$. A useful heuristic function must be both efficiently computable from the planning task, 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, critical trees, logical landmarks, and abstractions, have been translated to extremely useful heuristic functions, and different heuristics for classical planning can also be combined into their point-wise maximizing and/or additive ensembles.[1] Unfortunately, while some of these ideas have also been translated to classical planning with numeric state variables, the resulting heuristics do not appear useful for OSP. Approaching the need for effective heuristics for OSP, here we focus on abstractions for OSP, from their very definition and properties, to the prospects of deriving (admissible) abstraction heuristics.

Abstractions for OSP

The term "abstraction" is usually associated with simplifying the original system, factoring out details less crucial in the given context. In classical planning, the abstract t-graphs are required not to increase the distances between the (abstracted) states (Katz and Domshlak 2010b), and such "distance conservation" is in particular guaranteed by homomorphic abstractions, obtained by systematically contracting sets of states into single abstract states (Helmert, Haslum, and Hoffmann 2007). In turn, an additive abstraction in classical planning is a set of abstractions, inter-constrained to *jointly* not overestimate the state-to-state costs of the original task. As we now show, the concept of (additive) abstractions in OSP is very different, and, for better and for worse, has many more degrees of freedom than the respective concept in classical planning.

For $k \in \mathbb{N}^+$, by $[k]$ we denote the set $\{1, 2, \ldots, k\}$. Let $\mathcal{T} = \langle S, L, Tr \rangle$ be a tg-structure, and let $\mathcal{T}_i = \langle S_i, L_i, Tr_i \rangle$, $i \in [k]$, be a set of some tg-structures, each related to $\mathcal{T}$ via some state mapping $\alpha_i : S \to S_i$. Such a set of tg-structure/state-mapping pairs $\mathcal{AS} = \{(\mathcal{T}_i, \alpha_i)\}_{i \in [k]}$ is what is called an ***abstraction skeleton*** for $\mathcal{T}$ (Katz and Domshlak 2010b). Now, if $C_i \times U_i \times B_i$ is the performance measure space of $\mathcal{T}_i$, then $\mathbf{C} \times \mathbf{U} \times \mathbf{B}$, with $\mathbf{C} = \times C_i$, $\mathbf{U} = \times U_i$, and $\mathbf{B} = \times B_i$, is the *joint performance measure space of* $\mathcal{AS}$. That is, any choice of $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in \mathbf{C} \times \mathbf{U} \times \mathbf{B}$ induces a set of t-graphs $\{\langle \mathcal{T}_i, \mathbf{c}[i], \mathbf{u}[i], \mathbf{b}[i] \rangle\}_{i \in [k]}$. In turn, once $\mathcal{T}$ is associated with a performance measure (c, u, b), each joint performance measure $(\mathbf{c}, \mathbf{u}, \mathbf{b})$ of $\mathcal{AS}$ either does or does not constitute an (additive) abstraction of the t-graph $\Phi = \langle \mathcal{T}, c, u, b \rangle$. In Definition 1 we capture this relation at even a more refined level—with respect to a specific state of

[1] For a comparative survey and pointers to the literature, we refer the reader to Helmert and Domshlak (2009).

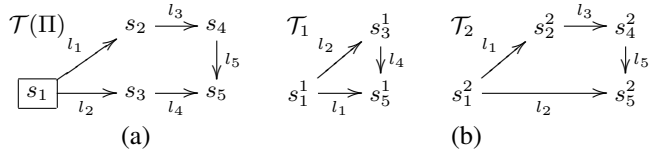

Figure 1: Illustration for our running example

interest in Φ—and we do that directly in terms of t-graphs induced by OSP tasks.

Definition 1 (Additive Abstraction)
*Let $\Pi = \langle V, A; s, c, u, b\rangle$ be an OSP task, $\mathcal{AS} = \{(\mathcal{T}_i, \alpha_i)\}_{i\in[k]}$ be an abstraction skeleton for $\mathcal{T}(\Pi)$, and $(\mathbf{c}, \mathbf{u}, \mathbf{b})$ be a joint performance measure for $\mathcal{AS}$. The set of t-graphs $\mathcal{A}_{(\mathbf{c},\mathbf{u},\mathbf{b})} = \{\langle \mathcal{T}_i, \mathbf{c}[i], \mathbf{u}[i], \mathbf{b}[i]\rangle\}_{i\in[k]}$ is an **(additive) abstraction for** Π, denoted by $\mathcal{A}_{(\mathbf{c},\mathbf{u},\mathbf{b})} \Subset_s \mathcal{AS}$, if*

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

i.e., when $h_{\mathcal{A}_{(\mathbf{c},\mathbf{u},\mathbf{b})}}(s)$ is an admissible estimate of $h^*(s)$.

In simple terms, a set of abstractions in OSP is constrained to jointly *not underestimate* the *value* that can be obtained from a *concrete state* of the original task within a given cost budget. For example, let $\mathcal{T} = \left\langle \{s_i\}_{i\in[5]}, \{l_i\}_{i\in[5]}, Tr\right\rangle$ in Figure 1a be a tg-structure of some OSP task Π with initial state s_1, and $\mathcal{AS} = \{(\mathcal{T}_1, \alpha_1), (\mathcal{T}_2, \alpha_2)\}$, with tg-structures $\mathcal{T}_1, \mathcal{T}_2$ as in Figure 1b and state mappings

$$\alpha_1(s_i) = \begin{cases} s_5^1, & i \in \{2,4\} \\ s_i^1, & \text{otherwise} \end{cases} \qquad \alpha_2(s_i) = \begin{cases} s_5^2, & i = 3 \\ s_i^2, & \text{otherwise.} \end{cases}$$

Let t-graphs $\Phi(\Pi) = \langle \mathcal{T}(\Pi), c, u, b\rangle$, $\Phi_1 = \langle \mathcal{T}_1, c_1, u_1, b_1\rangle$, $\Phi_2 = \langle \mathcal{T}_2, c_2, u_2, b_2\rangle$ be defined via label cost functions c, c_1, c_2 that associate all labels with a cost of 1, budgets $b = b_1 = b_2 = 2$, and state value functions u, u_1, u_2 that evaluate to zero on all states except for s_5, s_5^1, s_5^2, on which they respectively evaluate to one. Considering the state s_1 of Π, the optimal s_1-plan for Π is $\pi = \langle (s_1, l_2, s_3), (s_3, l_4, s_5)\rangle$ with $\hat{u}(\pi) = 1$. The optimal $\alpha_1(s_1)$-plan for Φ_1 is $\pi_1 = \left\langle (s_1^1, l_1, s_5^1)\right\rangle$ with $\hat{u}_1(\pi_1) = 1$, and the optimal $\alpha_2(s_1)$-plan for Φ_2 is $\pi_2 = \left\langle (s_1^2, l_2, s_5^2)\right\rangle$, with $\hat{u}_2(\pi_2) = 1$. Since $\hat{u}(\pi) \leq \hat{u}_1(\pi_1) + \hat{u}_2(\pi_2)$, $\mathcal{A} = \{\Phi_1, \Phi_2\}$ is an additive abstraction for Π.

Theorem 1 *For any OSP task $\Pi = \langle V, A; s, c, u, b\rangle$, any abstraction skeleton $\mathcal{AS}$ of $\mathcal{T}(\Pi)$, and any $\mathcal{A} \Subset_s \mathcal{AS}$, if the t-graphs of $\mathcal{A}$ are given explicitly, then $h_{\mathcal{A}}(s)$ can be computed in time polynomial in $||\Pi||$ and $||\mathcal{A}||$.*

The proof is straightforward: Let $\mathcal{A} = \{\Phi_i\}_{i\in[k]}$, with $\Phi_i = \langle \mathcal{T}_i, c_i, u_i, b_i\rangle$, be an additive abstraction for Π. For $i \in [k]$, let $S_i' = \{s' \in S_i \mid c_i(\alpha_i(s), s') \leq b_i\}$. Since $\mathcal{A}$ is given explicitly, computing shortest paths from $\alpha_i(s)$ to all states in $\mathcal{T}_i$, and thus computing S_i', can be done in time polynomial in $||\mathcal{A}||$ for all $i \in [k]$. If π_i is an optimal $\alpha_i(s)$-plan for Φ_i, then by Definition 1, $\hat{u}_i(\pi_i) = \max_{s'\in S_i'} u_i(s')$,

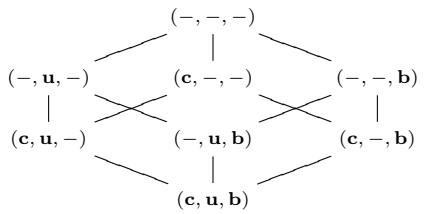

Figure 2: Fragments of restricted optimization over $\mathbf{A}(s)$.

and thus computing $h_{\mathcal{A}}(s) = \sum_{i\in[k]} \hat{u}_i(\pi_i)$ is polynomial time in $||\mathcal{A}||$.

While Theorem 1 is positive, it establishes only a necessary condition for the relevance of OSP abstractions to practice. Given an OSP task Π, and having fixed an abstraction skeleton $\mathcal{T}$ with a joint performance measure space $\mathbf{C} \times \mathbf{U} \times \mathbf{B}$, for each state of interest s, we should be able to automatically identify an abstraction that provides us with as accurate (aka as *low*) an estimate as possible. Let $\mathbf{A}(s) \subseteq \mathbf{C} \times \mathbf{U} \times \mathbf{B}$ be the subset of joint performance measures that constitute abstractions for Π. Note that $\mathbf{A}(s)$ is not a combinatorial rectangle in $\mathbf{C} \times \mathbf{U} \times \mathbf{B}$. For instance, consider t-graph $\Phi(\Pi)$, state s_1 of $\Phi(\Pi)$, 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, unit-cost functions, and two performance measures $(\mathbf{c}, \mathbf{u}, \mathbf{b}), (\mathbf{c}, \mathbf{u}', \mathbf{b}') \in \mathbf{C} \times \mathbf{U} \times \mathbf{B}$ being 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 $\mathbf{u}[1](s_5^1) = \mathbf{u}'[2](s_5^2) = 1$. It is easy to verify that $(\mathbf{c}, \mathbf{u}, \mathbf{b}), (\mathbf{c}, \mathbf{u}', \mathbf{b}') \in \mathbf{A}(s_1)$, yet $(\mathbf{c}, \mathbf{u}', \mathbf{b}), (\mathbf{c}, \mathbf{u}, \mathbf{b}') \notin \mathbf{A}(s_1)$.

We now proceed with considering a specific family of additive abstractions, reveal some of its interesting properties, and show that it contains substantial islands of tractability. We break down and approach the overall agenda of complexity analysis of abstraction-based heuristic functions under *fixation* of some of the three dimensions of $\mathbf{A}(s)$: If, for instance, we are *given* a vector of value functions $\mathbf{u}$ that is *known* to belong to the projection of $\mathbf{A}(s)$ on $\mathbf{U}$, then we can search for a quality abstraction from the abstraction subset $H_{(-,\mathbf{u},-)}(s) \subset \mathbf{A}(s)$, corresponding to the projection of $\mathbf{A}(s)$ on $\{\mathbf{u}\}$. As we show below, even some constrained estimate optimizations of this kind can be challenging. The lattice in Figure 2 depicts the range of options for such constrained optimization; at the extreme settings, $H_{(-,-,-)}(s)$ is simply a renaming of $\mathbf{A}(s)$, and $h_{(\mathbf{c},\mathbf{u},\mathbf{b})}(s)$ corresponds to a single abstraction $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in \mathbf{A}(s)$.

Partitions and Homomorphic Abstractions

With Definition 1 allowing for very general abstraction skeletons, in this work we focus on ***homomorphic abstraction skeletons***[2]: Given a tg-structure $\mathcal{T} = \langle S, L, Tr\rangle$, an

[2]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.

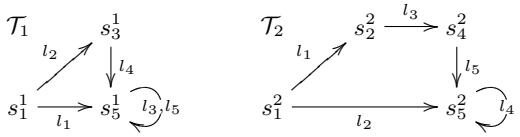

Figure 3: Homomorphic abstraction skeleton for $\mathcal{T}(\Pi)$ in Figure 1.

abstraction skeleton $\mathcal{AS} = \{\mathcal{T}_i, \alpha_i\}_{i\in[k]}$ of $\mathcal{T}$ is homomorphic if, for $i \in [k]$, $L_i = L$, and $(s, l, s') \in Tr$ only if $(\alpha_i(s), l, \alpha_i(s')) \in Tr_i$. In our running example, the abstraction skeleton depicted in Figure 1b is not homomorphic, but its slight extension as in Figure 3, *ceteris paribus*, is homomorphic. Furthermore, we focus on a fragment of additive abstractions

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

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. In what follows, by H_x^p we refer to $H_x \cap \mathbf{A}_p(s)$; e.g., $H^p_{(-,\mathbf{u},-)} = H_{(-,\mathbf{u},-)}(s) \cap \mathbf{A}_p(s)$. Given a t-graph $\Phi = \langle \mathcal{T}, c, u, b\rangle$, a homomorphic abstraction skeleton $\mathcal{AS} = \{\mathcal{T}_i, \alpha_i\}_{i\in[k]}$ of $\mathcal{T}$, and $\mathbf{c} \in \mathbf{C}$, we have $\mathbf{c} \in \mathbf{C}_p$ iff, for each label l in $\mathcal{T}$, $\sum_{i\in[k]} \mathbf{c}[i](l) \leq c(l)$. Similarly, $\mathbf{b} \in \mathbf{B}_p$ iff $\sum_{i\in[k]} \mathbf{b}[i] \leq b$, and (note the change in the direction of the inequality) $\mathbf{u} \in \mathbf{U}_p$ iff, for each state s in $\mathcal{T}$, $\sum_{i\in[k]} \mathbf{u}[i](\alpha_i(s)) \geq u(s)$.

Theorem 2 below establishes a "completeness" relationship between the sets $\mathbf{C}_p$ and $\mathbf{B}_p$, as well as an even stronger "completeness" of $\mathbf{C}_p$ and $\mathbf{B}_p$. In particular, it implies that, for all states s, the projections of $\mathbf{A}_p(s)$ 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.

Theorem 2 *Given an OSP task $\Pi = \langle V, A; s, c, u, b\rangle$ and a homomorphic abstraction skeleton $\mathcal{AS} = \{\mathcal{T}_i, \alpha_i\}_{i\in[k]}$ of $\mathcal{T}(\Pi)$,*

(1) for each action cost partition $\mathbf{c} \in \mathbf{C}_p$, there exists a budget partition $\mathbf{b} \in \mathbf{B}_p$ such that $\mathcal{A}_{(\mathbf{c},\mathbf{u},\mathbf{b})} \Subset_s \mathcal{AS}$ for all $\mathbf{u} \in \mathbf{U}_p$.

(2) for each budget partition $\mathbf{b} \in \mathbf{B}_p$, there exists an action cost partition $\mathbf{c} \in \mathbf{C}_p$ such that $\mathcal{A}_{(\mathbf{c},\mathbf{u},\mathbf{b})} \Subset_s \mathcal{AS}$ for all $\mathbf{u} \in \mathbf{U}_p$.

Proof: Let $\pi = \langle (s, a_1, s_1), (s_1, a_2, s_2), \ldots, (s_{n-1}, a_n, s_n)\rangle$ be an optimal s-plan in $\Phi(\Pi)$. Given that $\mathcal{AS}$ is homomorphic, let $\pi_1, \ldots, \pi_k$ be the projections of π on $\mathcal{T}_1, \ldots, \mathcal{T}_k$, respectively, that is, for $i \in [k]$, $\pi_i = \langle (\alpha_i(s), a_1, \alpha_i(s_1)), \ldots, (\alpha_i(s_{n-1}), a_n, \alpha_i(s_n))\rangle$.

(1) Let budget profile $\mathbf{b}^* \in \mathbf{B}$ be defined as $\mathbf{b}^*[i] = \sum_{j\in[n]} \mathbf{c}[i](a_j)$, for $i \in [k]$. First, note that $\mathbf{b}^* \in \mathbf{B}_p$ since

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

where $(*)$ is by $\mathbf{c}$ being an action cost partition, and $(**)$ is by π being an *s-plan* for $\Phi(\Pi)$. Second, for any $\mathbf{u} \in \mathbf{U}$, by the construction of $\mathbf{b}^*$, π_i is an $\alpha_i(s)$-plan for the t-graph $\langle \mathcal{T}_i, \mathbf{c}[i], \mathbf{u}[i], \mathbf{b}^*[i]\rangle$. Now, let $\mathbf{u} \in \mathbf{U}_p$, and for $i \in [k]$, let π_i^* be an optimal $\alpha_i(s)$-plan for that t-graph $\langle \mathcal{T}_i, \mathbf{c}[i], \mathbf{u}[i], \mathbf{b}^*[i]\rangle$. We have

$$\widehat{\mathbf{u}[i]}(\pi_i^*) \overset{(*)}{\geq} \widehat{\mathbf{u}[i]}(\pi_i) \overset{(**)}{\geq} \hat{u}(\pi), \tag{1}$$

where $(*)$ is by optimality of π_i^*, and $(**)$ is by π_i being the projection of π and $\mathbf{u} \in \mathbf{U}_p$. Therefore, $(\mathbf{c}, \mathbf{u}, \mathbf{b}^*)$ induces an additive abstraction for Π, that is, $\mathcal{A}_{(\mathbf{c},\mathbf{u},\mathbf{b}^*)} \Subset_s \mathcal{AS}$.

(2) Let cost function profile $\mathbf{c}^* \in \mathbf{C}$ be defined as $\mathbf{c}^*[i](a) = c(a) \cdot \frac{\mathbf{b}[i]}{b}$, for all actions $a \in A$, and all $i \in [k]$. First, we have $\mathbf{c}^* \in \mathbf{C}_p$ since $\mathbf{b} \in \mathbf{B}_p$ implies $1/b \sum_{i\in[k]} \mathbf{b}[i] \in [0, 1]$. Second, for any $\mathbf{u} \in \mathbf{U}$, by our construction of $\mathbf{c}^*$, π_i is an $\alpha_i(s)$-plan for the t-graph $\langle \mathcal{T}_i, \mathbf{c}^*[i], \mathbf{u}[i], \mathbf{b}[i]\rangle$. Following now exactly the same line of reasoning as the one around Eq. 1 above accomplishes the proof that $\mathcal{A}_{(\mathbf{c}^*,\mathbf{u},\mathbf{b})} \Subset_s \mathcal{AS}$ for any $\mathbf{u} \in \mathbf{U}_p$. □

Again, an important corollary of Theorem 2 is that, for all states s, the projections of $\mathbf{A}_p(s)$ 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. A priori, this property should simplify the task of abstraction optimization, and later we show that this is indeed the case. However, complexity analysis of abstraction optimization in most general terms is still problematic because OSP formalism is parametric in the representation of value functions. Hence, as a first step, we restrict our attention to a fragment of $\mathbf{A}_p(s)$ in which all abstract value functions are what we call 0-binary: A real-valued function f is a ***0-binary*** if it has image $\mathsf{img}(f) = \{0, v_f\}$ for some $v_f \in \mathbb{R}$. A set of 0-binary functions F is called ***strong*** if $v_f = v_{f'}$ for all $f, f' \in F$. On the one hand, 0-binary functions constitute rather a 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.

$\mathbf{A}_p(s)$ and 0-Binary Value Partitions

Important roles in what follows are played by a well-known Knapsack problem, as well as some tools from convex optimization. In a Knapsack problem $\langle \{w_i, \sigma_i\}_{i\in[n]}, W\rangle$, W is a weight allowance, $[n]$ is a set of objects, and each $i \in [n]$ has a weight w_i and a value σ_i. The objective is to find a subset $X \subseteq [n]$ that maximizes $\sum_{i\in X} \sigma_i$ over all subsets $X' \subseteq [n]$ with $\sum_{i\in X'} w_i \leq W$. By *strict Knapsack* we refer to a variant of Knapsack in which that inequality constraint is strict. Knapsack is NP-hard, 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 (Garey and Johnson 1978). 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 X by one of the weight-wise lightest objects in $[n] \setminus X$, until X cannot be expanded any further within W.

Let s be a state of an OSP task Π, $\mathcal{AS}$ be an (explicitly given) homomorphic abstraction skeleton of $\mathcal{T}(\Pi)$, and suppose that we *fix* a value partition $\mathbf{u} \in \mathbf{U}_p$. By Theorem 2, $H^p_{(-,\mathbf{u},-)}(s)$ is not empty, and thus we can try computing $\min_{(\mathbf{c},\mathbf{u},\mathbf{b}) \in H^p_{(-,\mathbf{u},-)}(s)} h_{(\mathbf{c},\mathbf{u},\mathbf{b})}(s)$. 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, despite that $H^p_{(-,\mathbf{u},-)}(s)$ is known to be non-empty, and so, too, are all of its subsets $H^p_{(-,\mathbf{u},\mathbf{b})}(s)$ and $H^p_{(\mathbf{c},\mathbf{u},-)}(s)$, finding an abstraction $(\mathbf{c},\mathbf{u},\mathbf{b}) \in H^p_{(-,\mathbf{u},-)}(s)$ is not necessarily easy.

In that respect, our first tractability results are for abstraction discovery within $H^p_{(-,\mathbf{u},-)}(s)$ 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 4, is on simultaneous selection of admissible pairs of cost and budget partitions. We also show how these results can be extended to pseudo-polynomial algorithms for *general* 0-binary value partitions.

Theorem 3 ($H^p_{(\mathbf{c},\mathbf{u},-)}(s)$ & strong 0-binary $\mathbf{u}$)
Let $\Pi = \langle V, A; s, c, u, b\rangle$ be an OSP task, $\mathcal{AS}$ be an explicit homomorphic abstraction skeleton of $\mathcal{T}(\Pi)$, and $\mathbf{u} \in \mathbf{U}_p$ be a strong 0-binary value partition. Given a cost partition $\mathbf{c} \in \mathbf{C}_p$, computing $h_{(\mathbf{c},\mathbf{u},\mathbf{b})}(s)$ for some abstraction $(\mathbf{c},\mathbf{u},\mathbf{b}) \in H^p_{(\mathbf{c},\mathbf{u},-)}(s)$ is polynomial-time in $||\Pi||$, $||\mathcal{AS}||$, and $||\mathbf{u}||$.

Proof: The proof is by reduction to the polynomial fragment of the Knapsack problem corresponding to all items having identical value. Let $\mathcal{AS} = \{\mathcal{T}_i, \alpha_i\}_{i\in[k]}$, and, given that $\mathbf{u}$ is a *strong* set of valued partitions, let $\mathsf{img}(\mathbf{u}[i]) = \{0, \sigma\}$. For $i \in [k]$, let w_i be the cost of the cheapest path in $\mathcal{T}_i$ from $\alpha_i(s)$ 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 algorithms for the single-source shortest paths problem. Consider now a Knapsack $\langle \{w_i, \sigma\}_{i\in[k]}, b\rangle$, with weights w_i being as above and value σ being identical for all objects. Let $X \subseteq [k]$ 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: for $i \in [k]$, $\mathbf{b}^*[i] = w_i$ if $x_i \in X$, and $\mathbf{b}^*[i] = 0$, otherwise.

What remains to be shown is that $(\mathbf{c},\mathbf{u},\mathbf{b}^*)$ actually induces an additive abstraction for Π, that is, $\mathcal{A}_{(\mathbf{c},\mathbf{u},\mathbf{b}^*)} = \{\langle \mathcal{T}_i, \mathbf{c}[i], \mathbf{u}[i], \mathbf{b}^*[i]\rangle\}_{i\in[k]} \in_s \mathcal{AS}$. Assume to the contrary that $\mathcal{A}_{(\mathbf{c},\mathbf{u},\mathbf{b}^*)} \notin_s \mathcal{AS}$, and let π be an optimal s-plan for Π. By the construction of our Knapsack problem and of $\mathbf{b}^*$, for each $i \in X$, there is a $\alpha_i(s)$-plan π_i in $\langle \mathcal{T}_i, \mathbf{c}[i], \mathbf{u}[i], \mathbf{b}^*[i]\rangle$ with $\hat{u}_i(\pi_i) = \sigma$. According to Definition 1, our assumption implies that $\hat{u}(\pi) > \sum_{i\in X} \hat{u}_i(\pi_i) = \sigma \cdot |X^*|$. On the other hand, from Theorem 2, there exists a budget partition $\mathbf{b} \in \mathbf{B}_p$ such that $\mathcal{A}_{(\mathbf{c},\mathbf{u},\mathbf{b})} \in_s \mathcal{AS}$. This budget partition induces a feasible solution $X' = \{i \mid w_i \le \mathbf{b}[i]\}$ for our Knapsack problem for which $\hat{u}(\pi) \le \sum_{i\in X'} \hat{u}_i(\pi_i) = \sigma \cdot |X'|$. This, however, implies $|X| < |X'|$, contradicting our assumption, and thus accomplishing the proof of $\mathcal{A}_{(\mathbf{c},\mathbf{u},\mathbf{b}^*)} \in_s \mathcal{AS}$. □

The construction in the proof of Theorem 3 may appear somewhat counterintuitive: while we are interested in minimizing the heuristic estimate of $h^*(s)$, the abstraction $\mathcal{A}_{(\mathbf{c},\mathbf{u},\mathbf{b}^*)}$ is selected via the value-maximizing Knapsack problem. However, a correct view of the situation would be that the selected triplet $(\mathbf{c},\mathbf{u},\mathbf{b}^*)$ provides us with the lowest estimate of $h^*(s)$ among all abstractions complying with $\mathbf{u}$ and $\mathbf{c}$, whose $\mathbf{A}(s)$ membership, aka admissibility, we know how to prove in polynomial time. Finally, while strong 0-binary value partitions are rather restrictive, finding an element of $H^p_{(\mathbf{c},\mathbf{u},-)}(s)$ 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 $H^p_{(\mathbf{c},\mathbf{u},-)}(s)$ with general 0-binary $\mathbf{u}$, running in time polynomial (also) in the *unary representation* of the budget b.

task	60%			80%			100%		
	blind	basic	$h_{\mathcal{A}}$	blind	basic	$h_{\mathcal{A}}$	blind	basic	$h_{\mathcal{A}}$
blocks-4-0	23	23	23	47	37	36	19	19	19
blocks-4-1	10	10	10	30	26	24	13	13	13
blocks-4-2	13	13	13	25	22	19	11	11	11
blocks-5-0	53	27	13	191	104	47	20	20	20
blocks-5-1	86	55	35	281	174	54	75	61	46
blocks-5-2	58	37	37	316	231	163	176	161	138
blocks-6-0	124	116	77	440	380	225	34	34	34
blocks-6-1	658	300	79	2415	1401	125	489	358	134
blocks-6-2	394	213	59	3501	2463	266	3135	2800	1549
blocks-7-0	390	331	225	7387	4220	1411	8370	6382	3865
blocks-7-1	6604	3265	480	42168	28862	602	12012	9582	2069
blocks-7-2	3171	2632	1709	28632	20533	8179	15914	12922	8010
blocks-8-0	12691	6116	323	165980	100970	1869	130780	96403	4892
blocks-8-1	46723	27582	16670	347914	252303	—	36504	31174	21358
blocks-8-2	9535	3931	145	89450	46822	154	1035	846	367
blocks-9-1	8651	8099	5069	907991	598110	—	734526	519569	—
blocks-9-2	21488	8754	820	925192	518385	913	1128285	813209	9390
driverlog-1	47	47	27	80	80	48	36	36	36
driverlog-2	18500	18500	6035	93238	93238	40489	4307	4307	2126
driverlog-3	1039	1039	377	5649	5649	905	231	231	231
driverlog-4	31741	31741	1786	272699	272699	22308	292	292	292
driverlog-5	71224	71224	8255	1373724	—	—	1635025	—	—
driverlog-6	8477	8477	419	62817	62817	2015	8279	8279	3034
driverlog-7	31293	31293	1421	619572	—	14709	107312	107312	—
logistics-4-0	15532	15532	12487	70845	70845	42452	65601	65601	52339
logistics-4-1	10255	10255	8109	50217	50217	29402	29187	29187	22727
logistics-4-2	3766	3766	2260	14198	14198	8947	2808	2808	2808
logistics-5-0	69013	69013	40087	311846	311846	—	291620	291620	—
logistics-5-1	6410	6410	2473	23175	23175	10941	2082	2082	2082
logistics-5-2	154	154	119	653	653	262	57	57	57
logistics-6-0	47896	47896	21915	231351	231351	—	81987	81987	58819
logistics-6-1	2246	2246	711	9661	9661	3668	474	474	474
logistics-6-2	47032	47032	21400	228617	228617	—	89914	89914	64010
logistics-6-9	31536	31536	15521	162325	162325	66593	11574	11574	10026
depots-1	137	137	137	265	261	261	233	233	233
depots-2	1460	1404	1210	5887	4986	3269	1518	1518	1518

Table 1: Expanded node statistics for optimal OSP with BFBB search on a set of IPC tasks, cast as OSP.

For a first test of the value that additive abstractions can bring to heuristic-search OSP, we have prototyped a simple planning system on the basis of Pyperplan, a lightweight planner written in Python[3]. Within that prototype, we have provided support for some basic pattern-database abstraction skeletons, action cost partitions, and abstraction selection in $H^p_{(\mathbf{c},\mathbf{u},-)}(s)$ for strong 0-binary value partitions as in the proof of Theorem 3. As best-first forward search algorithms such as A^* are not suitable for optimal OSP, we have implemented a best-first branch-and-bound (BFBB) search. This BFBB expands the nodes in the decreasing order of

[3] https://bitbucket.org/malte/pyperplan

their state values, with the ties being broken towards higher h-values, and then higher remaining budgets. As our heuristic estimates always upper-bound the true values achievable from states, if the h-value of a generated state is lower than the best state value encountered so far, then that generated state is pruned. The search terminates when the search frontier becomes empty, and the optimal plan is then extracted from the search node associated with the best-value state encountered so far.

Table 1 compares BFBB node expansions with three heuristic functions, tagged *blind*, *basic*, and $h_{\mathcal{A}}$, on set of IPC tasks that we cast as OSP by associating a separate value with each goal. With all three heuristics, the h-value of a node σ is set to 0 if the cost budget at σ is overconsumed. Otherwise, *blind* BFBB constitutes a trivial baseline in which $h(\sigma)$ is simply set to the total value of all goals. In *basic* BFBB, each goal is associated with its atomic (that is, single variable) projection abstraction, and $h(\sigma)$ 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. Finally, $h_{\mathcal{A}}$ is an additive abstraction heuristic that is selected from $H_{(\mathbf{c},\mathbf{u},-)}$ as in the proof of Theorem 3, with $\mathbf{c}$ being an ad hoc cost partition over atomic projections of the planning task onto goal variables, and $\mathbf{u}$ being a value partition that associates the value of each goal (only) with the respective atomic projection.

Each task was approached under three different budgets, which were 60%, 80%, and 100% of the minimal cost needed to achieve all the goals in the task. Despite the simplicity of the abstraction skeletons, the number of nodes expanded by BFBB with $h_{\mathcal{A}}$ is typically substantially lower than the number of nodes expanded by *basic* BFBB, with the difference sometimes reaching three orders of magnitude.[4] While this evaluation is still very preliminary, it testifies to the practical prospects of additive abstractions for OSP.

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 $\mathbf{c}$, thus buying more flexibility in selecting abstractions from $H^p_{(-,\mathbf{u},-)}(s)$ (and improving the accuracy of our estimates), while still remaining computationally tractable.

Definition 2 *Let $\Pi = \langle V, A; s, c, u, b\rangle$ be an OSP task, $\mathcal{AS}$ be a homomorphic abstraction skeleton of $\mathcal{T}(\Pi)$, and $\mathbf{u} \in \mathbf{U}_p$. By $\kappa_s(\mathbf{u})$ we refer to the largest value $v \in \mathbb{R}^{0+}$ such that, for <u>each</u> action cost partition $\mathbf{c} \in \mathbf{C}_p$, there exists a budget partition $\mathbf{b} \in \mathbf{B}_p$ with $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in H^p_{(-,\mathbf{u},-)}(s)$ and $h_{(\mathbf{c},\mathbf{u},\mathbf{b})}(s) \geq v$.*

Note that $\kappa_s(\mathbf{u})$ can be as low as 0 (and for us, "low" is good), even when, for any $0 \leq v \leq \max_{s\in S} \sum_{i\in[k]} \mathbf{u}[i](s)$, there exists some $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in H^p_{(-,\mathbf{u},-)}(s)$ with $h_{(\mathbf{c},\mathbf{u},\mathbf{b})}(s) \geq v$. In particular, note that $h(s) = \kappa_s(\mathbf{u})$ is at least as accurate as the estimate $h_{(\mathbf{c},\mathbf{u},-)}(s)$ from Theorem 3 for *any* fixed cost partition $\mathbf{c}$.

Theorem 4 ($H^p_{(-,\mathbf{u},-)}(s)$ & strong 0-binary $\mathbf{u}$) *Given an OSP task $\Pi = \langle V, A; s, c, u, b\rangle$, a homomorphic explicit abstraction skeleton $\mathcal{AS}$ of $\mathcal{T}(\Pi)$, and a strong 0-binary value partition $\mathbf{u} \in \mathbf{U}_p$, determining $\kappa_s(\mathbf{u})$ is polynomial-time in $||\Pi||$ and $||\mathcal{AS}||$.*

Let $\mathcal{AS} = \{\mathcal{T}_i, \alpha_i\}_{i\in[k]}$, and, given that $\mathbf{u}$ is a strong 0-binary value partition, let $\mathsf{img}(\mathbf{u}[i]) = \{0, \sigma\}$. Note that, for each $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in H^p_{(-,\mathbf{u},-)}(s)$, we have $h_{(\mathbf{c},\mathbf{u},\mathbf{b})}(s) = m\sigma$ for some $m \in \{0\} \cup [k]$. Our algorithm for determining $\kappa_s(\mathbf{u})$ is depicted in Figure 4, and its high-level flow is as simple as it gets: The for-loop of the algorithm decreasingly iterates over all the different estimates of $h^*(s)$ that can possibly come from the abstractions in $H^p_{(-,\mathbf{u},-)}(s)$, testing whether $\kappa_s(\mathbf{u})$ equals currently examined candidate estimate $m\sigma$. If the test (provided by the sub-routine always-achievable) is positive, then we are done. Otherwise, if the test fails for all $m \in [k]$, then $\kappa_s(\mathbf{u}) = 0$, which in particular implies that no state with value greater than 0 can be reached from s in Π with budget b.

The test of always-achievable for $\kappa_s(\mathbf{u}) = m\sigma$ is based on a certain linear program $\mathcal{L}_1(m)$, the semantics of which is captured by Lemma 1 below.[5] Informally, if $\mathcal{L}_1(m)$ is infeasible, then no cost partition over $\mathcal{AS}$ can provide us with the additive estimate of $m\sigma$, and this *independently of the budget allowance*; this can happen only if all σ-valued abstract states are simply unreachable from the respective abstractions of s in at least $k - m + 1$ tg-structures of $\mathcal{AS}$. Otherwise, if $\mathcal{L}_1(m)$ is solvable, then its solution establishes an action cost partition over $\mathcal{AS}$ that induces the *most costly achievement* of the additive estimate of $m\sigma$ using $\mathcal{AS}$, with the respective total cost being captured in the solution by a specific LP variable ξ.

Lemma 1 *Given a planning task $\Pi = \langle V, A; s, c, u, b\rangle$, let $\Pi/b' = \langle V, A; s, c, u, b'\rangle$. For all $m \in [k]$, if $\mathbf{x}$ is a solution of $\mathcal{L}_1(m)$, then*

$$\mathbf{x}[\xi] = \max_{\mathbf{c}\in\mathbf{C}_p} \min \left[b' \in B \;\middle|\; \begin{array}{l} (\mathbf{c}, \mathbf{u}, \mathbf{b}) \in H^p_{(-,\mathbf{u},-)}(s) \text{ w.r.t. } \Pi/b', \\ h_{(\mathbf{c},\mathbf{u},\mathbf{b})} \geq m\sigma \end{array} \right].$$

Otherwise, if $\mathcal{L}_1(m)$ is infeasible, then for no Π/b' there exists $(\mathbf{c}, \mathbf{u}, \mathbf{b}) \in H^p_{(-,\mathbf{u},-)}(s)$ with respect to Π/b' such that $h_{(\mathbf{c},\mathbf{u},\mathbf{b})} \geq m\sigma$.

The correctness of the algorithm with respect to Theorem 4 stems from Lemma 1: Suppose that the algorithm terminates within the loop, and returns $m\sigma$ for some $m > 0$. By the construction of the algorithm, $\mathcal{L}_1(m)$ is feasible and if $\mathbf{x}$ is a solution of $\mathcal{L}_1(m)$, then $\mathbf{x}[\xi] \leq b$. Lemma 1 then implies that, for *each* action 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 additive abstraction for Π and $h_{(\mathbf{c},\mathbf{u},\mathbf{b})}(s) \geq m\sigma$. If $m = k$,

[4] The runtime inefficiency of Python turned out to be an unfortunate obstacle: within the allowance of 30 minutes, some instances were solved by *blind* BFBB and were not solved even by *basic* BFBB, and this despite an extremely simple computation of *basic* h-values.

[5] The proof of Lemma 1 is omitted for lack of space.

input: $\Pi = \langle V, A; s, c, u, b\rangle$, $\mathcal{AS} = \{\mathcal{T}_i, \alpha_i\}_{i\in[k]}$,
strong 0-binary value partition $\mathbf{u} \in \mathbf{U}_p$
output: $\kappa_s(\mathbf{u})$

for $m = k$ **downto** 1 **do**
  if always-achievable(m) **then return** $m\sigma$
return 0

always-achievable(m):
  solve$(\mathcal{L}_1(m)) \mapsto$ *infeasible* / solution $\mathbf{x} \in dom(\mathcal{X})$
  if *infeasible* **or** $\mathbf{x}[\xi] > b$ **then return** false
    else return true

solve$(\mathcal{L}_1(m))$:
  set (5') to an arbitrary subset of constraints (5)
  loop
    set $\mathcal{L}'_1(m)$ to $\mathcal{L}_1(m)$, with constraints (5') instead of (5)
    ellipsoid-method$(\mathcal{L}'_1(m)) \mapsto$ *infeasible* / solution $\mathbf{x} \in dom(\mathcal{X})$
    if *infeasible* **return** *infeasible*
    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 return** $\mathbf{x}$
    extend (5') with constraint $\xi \leq \sum_{i\in[m]} \mathbf{b}[\tau(i)]$

Figure 4: An algorithm for computing $\kappa_s(\mathbf{u})$ for strong 0-binary value partitions $\mathbf{u} \in \mathbf{U}_p$ (Theorem 4).

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, Lemma 1 implies that there exists an action cost partition $\mathbf{c} \in \mathbf{C}_p$ for which no budget partition $\mathbf{b}$ of b will induce an additive abstraction for Π with $h_{(\mathbf{c},\mathbf{u},\mathbf{b})}(s) \geq (m + 1)\sigma$ (or, in case of infeasibility of $\mathcal{L}_1(m + 1)$, such a budget partition exists for no action cost partition at all.). Hence, by Definition 2, $\kappa_s(\mathbf{u}) < (m+1)\sigma$, and in turn, by the structure of $\mathbf{u}$, that implies $\kappa_s(\mathbf{u}) = m\sigma$. Finally, if the algorithm terminates after the loop and returns 0, then precisely the same argument on the basis of Lemma 1 implies $\kappa_s(\mathbf{u}) = 0$.

It remains now to specify our linear programs $\mathcal{L}_1(m)$ in detail, and analyze the complexity of solving them. Each such linear program is defined over variables

$$\mathcal{X} = \{\xi\} \cup \bigcup_{i\in[k]} \left[\{d(s')\}_{s'\in\mathcal{T}_i} \cup \{\mathbf{b}[i]\} \cup \bigcup_{a\in A} \{\mathbf{c}[i](a)\} \right], \quad (2)$$

constraints as in Eqs. 3-5, and the objective of maximizing the value of ξ. The roles of the variables in $\mathcal{L}_1(m)$ are as follows. Variable $\mathbf{c}[i](a)$ captures the cost to be associated with label a in the tg-structure $\mathcal{T}_i$. For state s' in $\mathcal{T}_i$, variable $d(s')$ captures the cost of the cheapest path in $\mathcal{T}_i$ from $\alpha_i(s)$ to s', *given* that the transitions (aka edges) are weighted consistently with the values of the variables $\mathbf{c}[i](\cdot)$. Variable $\mathbf{b}[i]$ captures the minimal budget needed for reaching in $\mathcal{T}_i$ a state with value σ from state $\alpha_i(s)$, *given* that, again, the transitions are weighted consistently with the variable vector $\mathbf{c}[i]$. Finally, ξ captures the minimal total cost of reaching states with value σ in *precisely* m t-graphs induced by $\mathcal{AS}$ under the joint performance measure $(\mathbf{c}, \mathbf{u}, \mathbf{b})$.

The constraints of $\mathcal{L}_1(m)$ are as follows. The first two sets of constraints in (3) come from a simple LP formulation of the single source shortest paths problem with the source node $\alpha_i(s)$: optimizing $\sum_{i\in[k]} \sum_{s'\in\mathcal{T}_i} d(s')$ under a fixed transition pricing $\mathbf{c}$ leads to computing precisely that. The third set of constraints in (3) establishes the costs of the cheapest paths in $\{\mathcal{T}_i\}$ from states $\alpha_i(s)$ to states valued σ, enforcing the semantics of variables $\mathbf{b}[i]$. Constraints (4) are the cost partition constraints, enforcing $\mathbf{c} \in \mathbf{C}_p$. Finally, constraints (5) enforce the aforementioned semantics of the singleton variable ξ.

$$\mathcal{L}_1(m):$$
$$\max \xi \quad \text{subject to}$$
$$\forall i \in [k]:$$
$$\begin{cases} d(s') = 0, & s' = \alpha_i(s) \\ d(s') \leq d(s'') + \mathbf{c}[i](a), & \forall (s'', a, s') \in \mathcal{T}_i \\ \mathbf{b}[i] \leq d(s'), & \forall s' \in \mathcal{T}_i, \mathbf{u}[i](s') = \sigma \end{cases}, \quad (3)$$
$$\forall a \in A : \sum_{i\in[k]} \mathbf{c}[i](a) \leq c(a), \quad (4)$$
$$\forall X \subseteq [k], |X| = m : \xi \leq \sum_{i\in X} \mathbf{b}[i]. \quad (5)$$

Note that, while the number of variables, as well as the number of constraints in (3) and (4), are polynomial in $||\Pi||$ and $||\mathcal{AS}||$, the number of constraints in (5) is $\binom{k}{m}$. Thus, solving $\mathcal{L}_1(m)$ using standard methods for linear programming is not practical. However, using the ellipsoid algorithm for linear inequalities (Grotschel, Lovasz, and Schrijver 1981), an LP with an exponential number of constraints can be solved in polynomial time provided that an associated separation problem can be solved in polynomial time. In our case, the separation problem is, given an assignment to the variables of $\mathcal{L}_1(m)$, test whether it satisfied (3), (4), and (5), and if not, produce an inequality among (3), (4), and (5) violated by that assignment. We now show how our separation problem for $\mathcal{L}_1(m)$ can be solved in polynomial time (see solve$(\mathcal{L}_1(m))$ in Figure 4) using what is called m-sum minimization LPs (Punnen 1992). As the number of constraints in (3) and (4) is polynomial, their satisfaction by an assignment $\mathbf{x} \in dom(\mathcal{X})$ can be tested directly by substitution. For constraints (5), 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 (5). Otherwise, we have our violated inequality $\xi \leq \sum_{i\in[m]} \mathbf{b}[\tau(i)]$.

From Strong to General 0-Binary Value Partitions

Recall that the polynomial result of Theorem 3 easily extends to a pseudo-polynomial algorithm for general 0-binary value partitions. It turns out that a pseudo-polynomial extension of Theorem 4 is possible as well, though it is technically more involved.

Theorem 5 ($H^p_{(-,\mathbf{u},-)}(s)$ **& 0-binary** $\mathbf{u}$)
Given an OSP task $\Pi = \langle V, A; s, c, u, b\rangle$*, a homomorphic explicit abstraction skeleton* $\mathcal{AS}$ *of* $\mathcal{T}(\Pi)$*, and a 0-binary*

input: $\Pi = \langle V, A; s, c, u, b\rangle$, $\mathcal{AS} = \{\mathcal{T}_i, \alpha_i\}_{i\in[k]}$,
0-binary value partition $\mathbf{u} \in \mathbf{U}_p$
output: $\kappa_s(\mathbf{u})$

let $0 < \epsilon < \min_{i\in[k]} \sigma_i$, $\alpha = 0$, $\beta = \sum_{i\in[k]} \sigma_i$
while $\beta - \alpha > \epsilon$ **do**
$v = \alpha + (\beta - \alpha)/2$
$\mathsf{solve}(\mathcal{L}_2(v)) \mapsto$ *infeasible* / solution $\mathbf{x} \in dom(\mathcal{X})$
if *infeasible* **or** $\mathbf{x}[\xi] > b$ **then** $\beta = v$
else $\alpha = v$
if $\alpha = 0$ **then return** 0; **else return** β

$\mathsf{solve}(\mathcal{L}_2(v))$:
set $(6')$ to an arbitrary subset of constraints (6)
loop
set $\mathcal{L}'_2(v)$ to $\mathcal{L}_2(v)$, with constraints $(6')$ instead of (6)
ellipsoid-method$(\mathcal{L}'_2(v)) \mapsto$ *infeasible* / solution $\mathbf{x} \in dom(\mathcal{X})$
if *infeasible* **return** *infeasible*
strict-Knapsack$(\{\mathbf{x}[\mathbf{b}[i]], \sigma_i\}_{i\in[k]}, \mathbf{x}[\xi]) \mapsto$ solution $X \subseteq [k]$
if $\sum_{i\in X} \sigma_i < v$ **then return** $\mathbf{x}$
extend $(6')$ with constraint $\xi \leq \sum_{i\in X} \mathbf{b}[i]$

Figure 5: An algorithm for computing $\kappa_s(\mathbf{u})$ for 0-binary value partitions $\mathbf{u} \in \mathbf{U}_p$ (Theorem 5).

value partition $\mathbf{u} \in \mathbf{U}_p$*, determining* $\kappa_s(\mathbf{u})$ *within* n*-digit precision*[6] *is polynomial-time in* $||\Pi||$, $||\mathcal{AS}||$, $\log(n)$*, and a* unary representation *of the budget* b *of* Π.

Let $\mathcal{AS} = \{\mathcal{T}_i, \alpha_i\}_{i\in[k]}$, and, for $i \in [k]$, $\mathsf{img}(\mathbf{u}[i]) = \{0, \sigma_i\}$. The algorithm for computing $\kappa_s(\mathbf{u})$ for inputs as in Theorem 5 is depicted in Figure 5. The flow of that algorithm bears some similarity to the one in Figure 4, yet it is different in many respects.

At the high-level, the algorithm performs a binary search over the hypothesis interval $[0, \sum_{i\in[k]} \sigma_i]$. The parameter ϵ serves as the "sufficient precision" criterion for termination; while any $\epsilon > 0$ can be used, adopting $\epsilon < \min_{i\in[k]} \sigma_i$ allows us to provide precision-independent answers in cases where $\kappa_s(\mathbf{u}) = 0$. At iteration corresponding to an interval $[\alpha, \beta]$, the algorithm attempts to solve a certain linear program $\mathcal{L}_2(v)$, testing the hypothesis $\kappa_s(\mathbf{u}) \geq v$, where v is the mid-point of $[\alpha, \beta]$. The test is positive if $\mathcal{L}_2(v)$ is feasible and the value $\mathbf{x}[\xi]$ in the respective solution $\mathbf{x}$ for $\mathcal{L}_2(v)$ indicates that, for the cost partition $\mathbf{c}$ induced by $\mathbf{x}$, there is a budget partition $\mathbf{b}$ that allows us to achieve the total (additive) estimate of at least v in t-graphs induced by $\mathcal{AS}$ under the performance profile $(\mathbf{c}, \mathbf{u}, \mathbf{b})$. If so, then the next hypothesis to test will be $\kappa_s(\mathbf{u}) \geq v'$, where v' is the midpoint of $[v, \beta]$. Otherwise, the next hypothesis corresponds to the midpoint of $[\alpha, v]$. To ensure admissibility of the estimate, upon termination of the loop, the estimate is set to β; the only exception is the case of the last (unexamined) interval being $[0, \epsilon]$, in which the estimate is safely set to 0. The correctness of the algorithm with respect to Theorem 5 stems from a lemma on $\mathcal{L}_2(v)$, which is identical to Lemma 1, *mutatis mutandis*.

The LPs $\mathcal{L}_2(v)$, $v \in \mathbb{R}^{0+}$, employed for testing hypotheses $\kappa_s(\mathbf{u}) \geq v$, are also defined over variables $\mathcal{X}$ as in Eq. 2, and are obtained from $\mathcal{L}_1(m)$ by replacing constraints (5) with constraints (6):

$$\mathcal{L}_2(v) : \max \xi \text{ subject to constraints (3), (4), and}$$
$$\forall X \subseteq [k] \text{ s.t. } \sum_{i\in X} \sigma_i \geq v : \quad \xi \leq \sum_{i\in X} \mathbf{b}[i]. \qquad (6)$$

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 $\{s_i\}_{i\in[k]}$ from states $\{\alpha_i(s)\}_{i\in[k]}$ in the respective k t-graphs induced by $\mathcal{AS}$ under the performance profile $(\mathbf{c}, \mathbf{u}, \mathbf{b})$ *such that* $\sum_{i\in[k]} \mathbf{u}[i](s_i) \geq v$. The new constraints (6) enforce this semantics of ξ (and thus the required max-min semantics of $\mathcal{L}_2(v)$). The number of constraints in (6) is $\Theta(2^k)$, and thus procedure *solve*$(\mathcal{L}_2(v))$ also employs the ellipsoid method with a sub-routine 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 pseudo-polynomial procedure for the *strict* Knapsack problem. Given an assignment $\mathbf{x} \in dom(\mathcal{X})$, its feasibility with respect to (3) and (4) can be tested directly by substitution. For constraints (6), let $X \subseteq [k]$ be an optimal solution to the strict Knapsack $\langle\{\mathbf{x}[\mathbf{b}[i]], \sigma_i\}_{i\in[k]}, \mathbf{x}[\xi]\rangle$.

- If the value $\sum_{i\in X} \sigma_i$ of X is *smaller* than v, then $\mathbf{x}$ satisfies *all* the constraints in (6). Assume to the contrary that $\mathbf{x}$ violates some constraint in (6), corresponding to a set $X' \subseteq [k]$. By definition of (6), $\sum_{i\in X'} \sigma_i \geq v$, and by our assumption, $\mathbf{x}[\xi] > \sum_{i\in X'} \mathbf{x}[\mathbf{b}[i]]$. That, however, implies that X' is a feasible solution for our strict Knapsack, and of value higher than that of presumably optimal X.
- Otherwise, if $\sum_{i\in X} \sigma_i \geq v$, then X itself provides us with a constraint in (6) violated by $\mathbf{x}$.

Summary

We defined and investigated fragments of additive abstractions for oversubscription planning. Along with revealing some significant islands of tractability, we exposed an interesting interplay between these abstractions and certain tools of combinatorial and convex optimization. Our empirical tests of the basic abstractions on a prototype system testified to the promise of the developed approach,. Our next steps will thus be to develop an efficient implementation of the entire framework, and to engage in further formal investigation of what is hard and what is tractable in the context of devising quality abstractions for oversubscription planning.

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

[6]The statement of Theorem 5 involves the precision of the estimate 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, $\kappa_s(\mathbf{u})$ can be determined precisely using a simplification of the algorithm we introduce to support the claim of Theorem 5. These details, however, are more of a theoretical interest; for reasonably small values of ϵ, in practice there will be no difference between estimates $h(s)$ and $h(s) + \epsilon$.

References

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

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

Coles, A. I.; Fox, M.; Long, D.; and Smith, A. J. 2008. A hybrid relaxed planning graph-LP heuristic for numeric planning domains. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS)*.

Edelkamp, S. 2002. Symbolic pattern databases in heuristic search planning. In *Proceedings of the International Conference on AI Planning and Scheduling (AIPS)*, 274–293.

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

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

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

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and 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)*, 1007–1012.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: WhatÕs the difference anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*.

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

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*.

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

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

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

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

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

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